2014

Keystroke and Touch-dynamics Based Authentication for Desktop and Mobile Devices

Zahid Ali Syed
West Virginia University

Keystroke and Touch-dynamics Based Authentication for Desktop and Mobile Devices

Zahid Ali Syed

Dissertation submitted
to the Benjamin M. Statler College of Engineering and Mineral Resources
at West Virginia University

in partial fulfillment of the requirements for the degree of

Doctor of Philosophy in
Computer Engineering

Bojan Cukic, Ph.D., Chair
Donald Adjeroh, Ph.D.
Thirimachos Bourlai, Ph.D.
Mark Culp, Ph.D.
Lawrence Hornak, Ph.D.
Katerina Goseva-Popstojanova, Ph.D.

Lane Department of Computer Science and Electrical Engineering

Morgantown, West Virginia
2014

Keywords: Behavioral biometrics, Habituation, Keystroke dynamics, Touch dynamics, Continual authentication, Active authentication, Mobile devices

ABSTRACT

Keystroke and Touch-dynamics Based Authentication for Desktop and Mobile Devices

Zahid Ali Syed

The most commonly used system on desktop computers is a simple username and password approach which assumes that only genuine users know their own credentials. Once broken, the system will accept every authentication trial using compromised credentials until the breach is detected. Mobile devices, such as smart phones and tablets, have seen an explosive increase for personal computing and internet browsing. While the primary mode of interaction in such devices is through their touch screen via gestures, the authentication procedures have been inherited from keyboard-based computers, e.g. a Personal Identification Number, or a gesture based password, etc.

This work provides contributions to advance two types of behavioral biometrics applicable to desktop and mobile computers: keystroke dynamics and touch dynamics. Keystroke dynamics relies upon the manner of typing rather than what is typed to authenticate users. Similarly, a continual touch based authentication that actively authenticates the user is a more natural alternative for mobile devices.

Within the keystroke dynamics domain, habituation refers to the evolution of user typing pattern over time. This work details the significant impact of habituation on user behavior. It offers empirical evidence of the significant impact on authentication systems attempting to identify a genuine user affected by habituation, and the effect of habituation on similarities between users and impostors. It also proposes a novel effective feature for the keystroke dynamics domain called event sequences. We show empirically that unlike features from traditional keystroke dynamics literature, event sequences are independent of typing speed. This provides a unique advantage in distinguishing between users when typing complex text.

With respect to touch dynamics, an immense variety of mobile devices are available for consumers, differing in size, aspect ratio, operating systems, hardware and software specifications to name a few. An effective touch based authentication system must be able to work with one user model across a spectrum of devices and user postures. This work uses a locally collected dataset to provide empirical evidence of the significant effect of posture, device size and manufacturer on user authentication performance. Based on the results of this strand of research, we suggest strategies to improve the performance of continual touch based authentication systems.

This work is dedicated to my family for the unwavering support, encouragement and understanding.

And to my wife, Sameera, my pillar of strength. You are, therefore I am.

Acknowledgements

- *To Dr. Bojan Cukic* – As an advisor, you have mentored me with patience and dedication. Thank you for helping me build upon my strengths and having faith in me. You went out of your way to provide me opportunities to develop professionally and were easy-going when I needed a break. Your genial nature helped make the Ph.D. so much more bearable and, in many instances, fun. Thank you for being an awesome advisor.

- *To Dr. Donald Adjeroh, Dr. Thirimachos Bourlai, Dr. Mark Culp, Dr. Katerina Goseva and Dr. Lawrence Hornak* –I deeply appreciate the effort you have put in to push me to a higher standard. Your observations and constructive criticism on my work have made me a better scientist and researcher. Your insightful comments have provided me a perspective from outside what I am typically accustomed to deal with and thus make this work more approachable.

- *To my lab mates* – Huihua Li, Mayra Sacanamboy, and Nathan Kalka: You have helped me out in one way or another during my education. Thank you for providing encouragement and support during this long (and sometimes masochistic) experience. And Sean: What a ride it has been! My heartfelt appreciation and my deepest thanks to you for being a wonderful, supportive colleague and a superb friend.

Contents

List of Figures

List of Tables

Chapter 1
Introduction

1.1 Motivation

Computer based access control systems rely on a variety of authentication procedures to restrict data access to legitimate users while preventing malicious users from accessing the same resources. The username-password scheme has been the de-facto method to protect and provide access to computer based systems such as e-mail, banking etc. The effectiveness of this method relies on the authorized user alone being aware of the username-password combination. Breaching this security layer requires little effort in today's age of pervasive social networks and computational devices. These systems offer weak protection and are easily circumvented with automated brute force attacks that attempt millions of character combinations per minute. Such attacks can be conveniently performed on systems that do not have a lock-out feature. Furthermore, once the password is known to unauthorized user, this user credential scheme has a 100% False Acceptance Rate (FAR). A variety of measures are used to to make the possibility of compromising credentials more difficult. These include secret question-answer combinations, text messaging an access code to a mobile device, frequently changing the password, etc. Commercial application of physiological biometrics (fingerprint, iris, face, hand) has proven difficult due to the greater cost, inefficiency, lack of infrastructure and the supervision required for user enrollment when deploying them. Due to this, the username and password based access is still relied upon as the most convenient access control method.

Desktop computers are accessed via the keyboard and mouse while touch screen based mobile devices such as cell phones and tablet computers use touch presses and touch gestures as sources as input. Thus, keystroke dynamics and touch dynamics are soft-biometric based authentication tools and can be used to create an effective multi-modal system for continual authentication in a multi-device environment. In modern times, a single person uses multiple computer devices such as a desktop computer, a cellphone and a tablet computer, etc. Each modality presents certain challenges to understand usage patterns, and isolate factors that affect user model development in order to better authenticate a user.

1.2 Goal

The goal of this work is to develop practical strategies to improve user authentication within keystroke dynamics and touch dynamics based authentication systems. This work has been carried out on desktop computers and mobile devices. With respect to desktop and mobile computers, our focus has been to identify and study the effect of habituation. As a user learns to type a password or use a mobile device application, his/her keystroke and touch dynamics profile

changes. This 'habituation' has a significant effect on user behavior, profile modeling and, thus, authentication performance. Our goal is to empirically study this effect and its impact on authentication. We also analyze the intra-user and inter-user variations caused by habituation on usage patterns.

Touch dynamics based authentication systems for mobile touch screen computers must also take additional factors into account to develop an accurate user profile. This is due to the portable nature of mobile devices such as tablets and cell phones. The factors include user's posture, device type, size, orientation, type of app, gesture direction, etc. Creating a user profile using *apriori* knowledge of such factors may allow for a more accurate representation of a user model. It also leads to better authentication while requiring less data for user model creation. Our goal is to use an experimental data collection to empirically study the effects of a number of such factors upon a user touch dynamics profile. The empirical results from this study provide information on which factors have a significant effect on improving accuracy in determining impostors. This study has immediate practical benefits to develop strategies for better authentication through touch dynamics. Our work on touch dynamics authenticates users continually, i.e. the user is authenticated as they are using the device.

1.3 Contributions

This work provides the following set of original contributions:

1. Analysis of the effect of user habituation in keystroke dynamics on a user's keystroke dynamics profile, its effect on inter-user profile separability with time, and its impact on authentication performance. We show that user habituation has a statistically significant impact upon the user profile and the performance of keystroke dynamics based authentication schemes. Based on our findings, we test several training models and conclude that the best model to improve the accuracy of keystroke dynamics authentication systems is one that retrains on the most recent set of keystroke entries.

2. Demonstration that event sequences are an effective attribute for use in keystroke dynamics based authentication systems. The event sequence is the temporal sequence of all key-press and key-release events performed to type a string. This includes the key-press and key-release events of character keys and of special keys (Caps Lock, Left Shift, Right Shift, etc) that are used to modify the character key output. In contrast to the traditional approach in literature of ignoring these variations, we show using empirical analysis that including variations in event sequences in keystroke dynamics based authentication systems leads to better performance and reliability. We also demonstrate that event sequences possess discriminatory information that is independent of typing proficiency. This is in contrast to currently existing keystroke dynamics based attributes, all of which rely on inter-user variations in typing proficiency.

3. Demonstration that the user's posture, the device size and the device manufacturer have a significant impact on the authentication performance of a touch-based authentication system. We show that the attributes used in current state-of-the-art touch-based authentication systems lead to a user model that is incapable of providing constant, reliable performance when any of the above-mentioned three factors are changed.

1.4 Organization

The remainder of this work is organized as follows. Chapter 2 provides a summary of related work regarding keystroke dynamics and touch dynamics. Chapters 3 and 4 describe specific contributions in the areas of keystroke dynamics and touch dynamics, respectively. Finally, Chapter 5 concludes the work by providing a summary of the accomplishments as well as future directions for research.

Chapter 2
Related Work

2.1 Keystroke Dynamics

2.1.1 Introduction

The username and password authentication is the de-facto standard for computer access control due to the low cost of implementation and infrastructure. The effectiveness of this system hinges on the assumption that only the legitimate, genuine user knows his or her credential. This type of authentication can be circumvented through either brute force or other more complicated attacks [78]. A variety of measures are currently used to harden the credential checking process, including visual captchas, user-specific questions, account to device or location association, regular password changes etc. With the advent of social networks, user specific information, possibly hinting on the content of weak passwords became easier to obtain [31] while IP/MAC addresses in two way authentication schemes can be faked using off-the-shelf software [18], making password authentication breaches more common.

Adoption of traditional biometric authentication techniques for computer access control (fingerprint, iris or face recognition), on the other hand, can be an expensive proposition. High accuracy of biometric authentication typically correlates with the use of costly sensors, but sensor interoperability is not guaranteed [62]. Thus, enhancements to low cost password-based authentication, which provides an additional level of trustworthiness are more appealing.

Keystroke dynamics is a behaviorial biometric with effective performance potential [9]. It evaluates the typing behavior of users by measuring the duration of each key press, the latency between successive key presses etc. These time periods are called the hold and delay times, respectively. Hold times will always exhibit positive values as a finite amount of time is required to press a key, while delay times may be positive or negative. A negative delay time occurs when a user presses the succeeding key prior to releasing the current key.

Keystroke dynamics based systems can be deployed under either a fixed text or a free text scenario. Fixed text analysis is constrained to pre-determined content and is effective for building static user authentication systems (such as username and password login). Free text, on the other hand, can be used to develop continual authentication systems[80] that determine whether an impostor has taken over another user's session.

2.1.2 Terminology

When a key on a keyboard is pressed down, a make code for the key is transmitted to the device while encoded as a hex value. It contains information on the key pressed, and other system flags such as any modifier keys used (shift,

control, alt, etc). When the key is released, a corresponding break code is sent to the device. This information received at the lowest level is used to define higher level events called the KeyDown event and the KeyUp event. KeyDown event refers to the time when the key is engaged while KeyUp refers to the time when it is disengaged. Based on these two events, derivative attributes are created such as hold time, and delay time. Hold time refers to the period of time between the KeyDown event and the corresponding KeyUp event. Hold time is also referred to as dwell time in literature. Delay time, also referred to as flight time or latency, is the period between two consecutive key events. Delay time may be positive or negative. It is negative if the succeeding key was pressed before releasing the preceding key.

Keystroke dynamics is restricted to using the duration of keypresses for user profile modeling. There have been attempts to create pressure sensitive keyboards [70] that senses the force level on every depressed key and implement this technology in keystroke dynamics [20]. However, efforts in this direction have been mainly for research purposes [58, 34, 49] and pressure sensitive physical keyboards have not yet been produced commercially.

2.1.3 Current Research

Utilizing typing behavior for user authentication is a well explored field. For example, by the mid 19^{th} century, it was understood that telegraph operators had unique "tapping" signatures. By the late 1970s, SRI International had developed the first hardware based implementation of keystroke dynamics based systems. In 1984, NIST conducted a study that found the technology to be "98% effective". Starting in the early 2000s there has been a continued growth in the understanding of keystroke dynamics [45].

Only in the past 10 years have researchers explored the feasibility of creating frameworks for keystroke - based authentication using machine learning techniques. Of particular interest is the exploration of various classifier methods for authentication. The Bayes classifier [16], hidden Markov [15], Gaussian mixture [36] and k-nearest neighbor models [37] have been explored, as have different distance measures in [35, 46]. More recently neural networks [60, 61, 52], Support Vector Machines [72, 81], and ensemble learning through random forests [6, 73] have also been studied.

The physical characteristics of keystroke dynamics, such as keyboard layout [76], habituation [73, 4, 43, 44] and model update techniques [44, 43, 4] received increased attention. In [73] the authors theorized that habituation was dependent on the type of password and could be leveraged for building more effective classifiers through the reduction of Equal Error Rates (EER). However, the study was confounded by the presence of larger training samples and lacked a rigorous statistical analysis of the effect of habituation over time. Authors of [4, 42] noted that a mere increase from 5 to 10 samples for training could have a marked effect on reducing error rates. In [44, 43, 4] the authors contended that updating the user model upon successful authentication could enable better classifier performance. This work offers credible statistical evidence to this

research direction.

In [4] the authors noted that the classification algorithm, the volume of training data, and the update technique have a strong impact on performance, while the feature set does not. Our work demonstrates similar findings. However, we contend that the volume of training is confounded by the variability of the prior data. In this paper we showed that the reduction in variance indicates that the best performance can be obtained by using recent keystroke entries for modeling. Moreover, the feature sets in our long and short passwords demonstrated that the type of password playes a key role in the classifier performance. When assigned more complex credentials, the users were easier to differentiate.

The selection of passwords also plays a critical role in keystroke dynamic research. In [4, 46] the authors assigned a single password to all users. In [4, 46] the users were allowed to select their own passwords. While this represents a more realistic scenario, statistical studies will be confounded by the varying lengths and complexity of passwords. A more thorough history of keystroke dynamic based authentication can be found in [5].

In [71], Sim and Janakiraman showed that the digraph pattern changes when it occurs in different words. A distance measure between two typing samples based on the relative typing speed of trigraphs was introduced in [8].

In [32], Gunetti and Claudia evaluated the use of free text in detecting impostors. The experiment was setup by having forty volunteers type 15 samples of text each. These volunteers served as the legal users of the hypothetical system. Another 165 people were then used as impostors and their typing samples were gathered over the course of 6 months. This study reported a False Acceptance Rate (FAR) of 5% and a False Rejection Rate (FRR) of 0.005%. An extended approach to incorporate digraphs and trigraphs is proposed. Another distance measure to consider difference of n-graph's timing information in two typing samples is also introduced.

In [61], Obaidat and Saudon used 15 volunteers over a period of 8 weeks to conduct the experiment. Each user logged in using their user ID which had an average length of 7 characters. The same users also acted as impostors by trying to login using the other users' IDs. The keystroke data was partitioned into a training set and a testing set and run through various pattern recognition and neural network based classifiers. The data was analyzed by using hold times and delay times datasets separately and also by combining them. According to the authors, the combined hold and delay time-based dataset consistently performed better than using hold and delay datasets separately. Furthermore, the authors show that the best performing neural network classifiers gave lower FAR and FRR values than the best performing traditional pattern recognition techniques such as K-means, Bayes classifier and Cosine measure.

In [55], the keystroke data was collected over a period of 7 weeks with almost all of the participants being familiar with computers. The subjects were asked to type in a phrase that was displayed on the interface. For the free text section of the experiment, the subject typed sentences of their own accord. The total number of participants used in the final data analysis was 31. The dataset was divided into training and testing datasets. The classification algorithms used

included Euclidean distance measure, non-weighted probability and weighted probability measure. The weighted measure algorithm was used to weigh the more commonly occurring digraphs such as 'er', 'th', 're' as compared to less common digraphs. This form of classification led to a 10% increase in the classification accuracy. The authors also note that they found that the mean typing speed of digraphs that were typed only with the left hand such as 'er', 're', 'es', 'we' were lower for left-handed users as compared to when typed by right handed users.

Villani et. al. [77] focused on the use of long-text input for gathering keystroke data. The two variables that were analyzed in this study were the type of input keyboard (laptop or desktop) and the type of text input (free text or copying a given string). The Nearest Neighbor classification algorithm was used to perform the analysis. The experiment was performed on 118 subjects. Data collection consisted of entering data on a desktop or a laptop. Each scenario had two activities where the subject could either copy the given text or type free text. For data acceptance, a subject had to participate in at least two of the four possible activities.

Based on the results, the authors note that the accuracy of identifying a subject is highest when a single keyboard is used by the subject during training and testing phases. Laptop accuracies were also reported to be higher than desktop accuracies, presumably due to the personal nature of laptops. The authors also note that the accuracy of free text input is lower than when copying a given string. Accuracy also decreased when subjects used different keyboards for enrollment and testing (Laptop for enrollment and desktop for testing or vice versa). The authors did not detect any significant difference between desktop-desktop or laptop-laptop training-testing phases and neither did they detect any difference between a copy task and free text on desktops. However, there was a distinct performance downgrade in user classification when copy tasks and free text entry were performed on the same keyboard and were then used as training and testing datasets respectively. According to the authors, this indicates that user patterns change based on the mode of typing (free text or copy). The accuracy decreased further when different keyboard types and different input modes were used, which suggests that the type of keyboard used in the experiment affects the performance of the classifier.[77]

Another aspect of keystroke analysis that has been researched pertains to the use of keystroke digraphs, trigraphs, and n-graphs in user identification. Bergadamo et. al. achieved a False Acceptance Rate (FAR) of 0.007% and a False Reject Rate (FRR) of 4.09% in [8]. Free text analysis for computer based systems has been performed by [32, 71]. One of the reasons for this is that behavioral biometrics may change according to the physical environment of the user. Monrose and Rubin, while working with free text, achieved a classification accuracy of about 23%. Research work related to the study of keyboard characteristics has also been performed by [76, 70].

2.1.4 Contributions to keystroke dynamics

In this work, we analyse the effect of user habituation in keystroke dynamics on a user's keystroke dynamics profile, its effect on inter-user profile separability with time, and its impact on authentication performance. We show that user habituation has a statistically significant impact upon the user profile and the performance of keystroke dynamics based authentication schemes. Based on our findings, we test several training models and conclude that the best model to improve the accuracy of keystroke dynamics authentication systems is one that retrains on the most recent set of keystroke entries.

We also demonstrate that event sequences are an effective attribute for use in keystroke dynamics based authentication systems. The event sequence is the temporal sequence of all key-press and key-release events performed to type a string. This includes the key-press and key-release events of character keys and of special keys (Caps Lock, Left Shift, Right Shift, etc) that are used to modify the character key output. In contrast to the traditional approach in literature of ignoring these variations, we show using empirical analysis that including variations in event sequences in keystroke dynamics based authentication systems leads to better performance and reliability. We also demonstrate that event sequences possess discriminatory information that is independent of typing proficiency. This is in contrast to currently existing keystroke dynamics based attributes, all of which rely on inter-user variations in typing proficiency.

The research on keystroke dynamics presented in this work offers strategies for training better keystroke-dynamics based authentication systems. Furthermore, we demonstrate how the definition of keystroke dynamics can be expanded to include the user's key-preference behavior in addition to traditional typing rhythm behavior.

2.2 Pointer dynamics

2.2.1 Definition

Pointer-based recognition is a behavioral biometric that analyzes the usage pattern of a computer's pointing device to verify the identity of an individual. A pointing device is used to move the cursor within the Graphical User Interface of a computer or another display device. It can also be used to select interface elements or perform actions based on the control elements available. Computer pointing devices have been investigated as a form of behavioral biometric only recently. The advantage of this modality is the widespread applicability, as it only requires the hardware present in most computing systems. In the literature, the most common examples of pointer-based recognition refer to mouse dynamics. Current extensions to touch sensitive devices lead to touch dynamics.

The development of algorithms that can determine the identity of a user starts with the identification of the device control elements. A computer mouse is typically used as a pointing device on desktop and some laptop computers. A computer mouse controls consist of two or three buttons and a movement-

tracking device such as a rolling ball or a laser. At the atomic level, two types of events characterize any pointing activity: mouse movement and button events. The former refers to the actions of moving the screen pointer while the latter refers to mouse button's presses and releases. Software can trap and record atomic mouse events. Events are associated with attributes, e.g., the event type, timestamp(s), and cursor coordinates. This information collectively forms the mouse dynamics data.

Atomic events seem to be too detailed to be effective for biometric analysis. The literature suggests they be aggregated into higher-level abstractions to obtain meaningful representations of user behavior. The abstractions include gestures, such as drag-and-drop, move-and-click, right-click, left-click, and simple mouse movement [1]. Gestures aggregate usage patterns. Statistical analysis of associated measures allows the development of user models and their comparison.

In computing devices with touch screens interface interactions require touch gestures, typically performed with the fingers. These touch gestures require either one or two fingers to execute. Examples of single finger gestures include touch, double touch, and swipe. Two finger gestures include pinch open and pinch close. At the atomic level, the a touch gestures are recorded by the device as the series of touch presses. Typical atomic data includes the location of the press, the screen area pressed by the finger at that point, and the timestamp of the event. Higher-level abstractions provide more meaningful information such as the gesture type, the gesture velocity and pressure at various intervals, deviations from "a perfect gesture", etc. These statistics can be aggregated over a period of time to form a user model. Most touch screens used in smart phones do not have the technology to capture touch-pressure information. However, the relative measure of pressure can be computed indirectly, by measuring the area touched by the finger.

2.2.2 Mouse Technology

Modern commercially available mice either use mechanical or optical means to detect mouse movements. A mechanical mouse utilizes a ball located at the bottom of the mouse. As the user moves the mouse, the ball rotates in the same direction. Using mechanical means, the ball's movement is translated to pulses of infrared pulses that are detected by an infrared sensor. By using two pairs of pulses and sensors, orthogonally on the X and Y coordinates, the pulses are interpreted by a processor to calculate the speed and direction of the mouse's movement. Optical mice use an LED or a laser for tracking movements. This is performed by casting light on the surface that the mouse is being moved over. A CMOS sensor captures images of the light that scatters off the surface. A digital signal processor samples the images from the CMOS sensor. Using a signal processing algorithm to compare the changes in the images captured by the CMOS sensor, the digital signal processor calculates the movement direction and speed of the mouse. Due to their high sampling rate, optical mice have a higher tracking resolution compared to mechanical mice. They also have lower

failure rates, are more resistant to dirt accumulation and do not require a special mouse pad to work. Due to these reasons, they have are used more commonly than mechanical mice.

2.2.3 Usage Scenarios

Mouse Dynamics-based Authentication (MDA) systems have been used in two types of biometric scenarios: static authentication and continual authentication. Static MDA systems authenticate a user at a specific time such as at the time of login. Such a system needs to offer a definitive authentication outcome in a short time period. This usually requires the user to enter a predetermined input sequence, such as trace a pattern [64], or offer an electronic signature unique to the user. Due to the constrained nature of mouse activity, static MDA systems are easily implemented and computationally inexpensive.

On the other hand, continual MDA systems offer repeated authentication of a user's identity as he or she continues to operate the pointing device. In fact, authentication is continually performed throughout the usage session. In this scenario, the user's mouse movements are unconstrained. Mouse dynamics data is collected by a background process, with or without user's cooperation or his/her awareness. The authentication system has no control over the form, sequence and intensity of user activity. While the lack of constraints inspires uninhibited behaviors thought to offer increased uniqueness, the lack of activity may lead to periods in which authentication is not feasible. When mouse dynamics activity is present, incremental data is analyzed and compared to the stored user model [1]. Due to incremental nature of data collection, continual MDA systems may offer authentication decisions with varying levels of certainty over time. Accurate decisions require longer periods of mouse dynamics data monitoring, typically in terms of minutes or tens of minutes. Continual authentication systems are computationally more expensive than the static ones.

2.2.4 Feature Representation

Most MDA systems developed so far rely on machine learning methods for model development and comparison. Therefore, users are required to enroll. The enrollment includes a period of time needed for model building, leading to the establishment of a signature, which is stored in a model. Such a model represents the basis for user authentication, in which the user's mouse dynamics data is captured and compared to the model.

Based on the features used for authentication, MDA systems can be categorized into three types [48]:

1. Trajectory-based authentication is suitable for static MDA systems. The user is asked to follow a standard pattern such as a signature or an outline of a drawing that serves as a mouse dynamics based "password". In this type of system, various trajectory-based measures are calculated at sampling points on the pattern [10, 12]. These may include angle, distance

time and between predefined sample points, etc. The similarity between the enrolled signature and the input mouse data is calculated to authenticate the user. Due to the narrow set of patterns used for enrollment and verification, trajectory-based approaches by themselves are not suitable for continual authentication systems.

2. Feature-based authentication incorporates additional attributes of mouse data not included in trajectory-based systems. In addition to the tracking of precise pointer coordinates, this approach includes derived second order measures of mouse movements, such as speed, acceleration, angular velocity, and curvature [64]. These features are computed from the atomic mouse events. Feature-based authentication systems are suitable for both static and continual authentication.

3. Behavior-based approaches include mouse dynamics data collected over a longer time periods. The lengthy mouse dynamic data vectors allow the computation of various cumulative measures and statistical parameters, for example, the average speed of the pointer, movement direction histogram, traveled distance, etc. This information is used not only to compare against the user's enrolled signature but, in some cases, to periodically update the model. Behavior-based analysis allows similarity score to be applied to biometric identification (one to many matching) [1], in addition to authentication. A considerable amount of time may be needed to accumulate enough data to identify a user. Nevertheless, continual authentication systems in which mouse movements are unrestricted and the number of users who participate in identification is large require the added complexity of behavior-based mouse dynamics approaches.

2.2.5 Classification

A variety of algorithms have been proposed for the purpose of comparing the mouse data captured during the verification stage with the user model. These approaches include neural networks [1, 64], distance metrics [48, 65, 57, 12], decision trees [48, 68], support vector machines [81, 52, 76], and data distribution models [10].

One of the problems with analyzing the accomplishments in the field of mouse dynamics authentication is the wide variety of experimental setups in literature. Table 1 compares the experimental results from the selected works on static and continual authentication scenarios with mouse dynamics. The table shows that the reported performance of MDA systems varies considerably, but so do the experiment setups. The differences in experimental setups are related to the number of test subjects, the restrictions of the devices used by the subjects, the type of mouse signatures in static authentication, the features chosen for classification, the nature of impostor data, and/or the algorithm used for classification. At this time, there is no agreement amongst the researchers with respect to relevant test scenarios or the best performing classification approaches. Therefore, claimed performance results reported directly from the

literature in Table 1 should not be interpreted as a clear indication of quality of a classification approach.

For example, Jorgensen and Yu [41] compared two previously proposed classification approaches by Ahmed and Traore [1] and Gamboa and Fred [26] under the same scenarios and in a unique environment. They found that when environmental variables uncontrolled in the previous studies (such as the length of enrollment time and the devices used by test subjects) were controlled, the error rates changed considerably. This seems to indicate the uncontrolled variables contribute to the distinctiveness of users, thereby artificially increasing the algorithm performance. While Jorgensen's study used only 17 subjects, the results highlight the problems that are rather common for an emerging biometric field.

Shen et al. conducted a comparison study of 8 classification algorithms using the dataset from a controlled data collection [69]. They found that certain distance metrics (Nearest Neighbor and Mahalanobis) offer good authentication performance. However, it is not clear whether these same distance metrics would perform well with a different feature set. Furthermore, noise is an inherent problem in behavioral biometrics. Further research is needed to determine whether the best performing algorithms are robust to noise. In a related behavioral biometrics domain (keystroke dynamics) it has been shown that user habituation affects classifier performance [73]. It is likely that this phenomenon affects the performance of mouse dynamics based authentication too. Such confounding factors make it difficult to attribute experimental results solely to the nominal user behavior at this time.

The metrics commonly used to determine the efficacy of biometric authentication approach are False Accept, False Reject and Equal Error Rates. The parameter important for pointer based biometrics is the time needed for authentication. We also report the number of users in the study. Table 1 indicates that while research claims a significant increase in the performance of MDA systems, the performance and reliability necessary for stand-alone deployment in real-world systems does not appear to be within reach yet.

Table 1: Overview of selected works on static and continual authentication using mouse dynamics

Work	Type	Features used	Algorithm	Authentication time per user	# of users	Error Rates	
						FRR	FAR
Hashia et al (2005) [33]	Static	Pointer speed, angle and distance of deviation from shortest path between 2 points	Outlier detection	20 secs	15	15%	15%
Bours and Fullu (2009) [12]	Static	Velocity vectors for each segment of mouse movement while traversing a maze	Levenshtein distance	Not reported	28	27%	27%
Sayed et al (2013) [64]	Static	8 gestures types characterized by 12 features such as velocity, curvature, acceleration etc.	Neural network	26.9 secs	39	4.59%	5.26%
Schulz (2006) [65]	Cont.	Mouse movement curve features: length, number, curvature, inflection features and straightness	Euclidean distance	Not reported	72	24.30%	24.30%
Nakkabi et al. (2010) [57]	Cont.	39 features based on 4 types of movements: mouse-move, drag-and drop, point-and-click, silence	Fuzzy classification	17 mins	48	0.36%	7.78% 2.75%
Shen et al (2012) [68]	Cont.	Click elapsed time, movement speed, movement acceleration, relative position of extreme speed	One-class SVM (Best performing), k-nearest neighbor, neural network	10 mins	28	9.45% 3.39%	7.78% 2.75%

2.2.6 Application and Challenges

Mouse dynamics has the potential to be used as a stand-alone biometric. The more likely context, currently, is its inclusion in multi-modal systems. Fusion of mouse dynamics data with other biometric modalities provides a faster, more reliable solution [74]. Research studies fail to test MDA systems with challenging (non zero-effort) impostor data. Reducing the authentication time remains a significant challenge in continual authentication scenarios. A long authentication time may allow a malicious user sufficient time to penetrate the system.

Nakkabi et al. achieved an Equal Error Rate (EER) that matches the European standards for security in continual authentication systems [57]. This result came at the cost of a long time needed for authentication (˜17 minutes) using 48 test subjects who posed as impostors for one another.

More generally, current experiments suffer from a small number of test subjects and limited impostor data that comes either from the users included in the study or is created artificially. A publicly available realistic dataset would significantly help advance the field. Such a dataset would need to minimize various confounding factors, such as those related to different pointing devices, screen sizes and resolutions, mouse acceleration settings, familiarity of users with applications, etc.

2.2.7 Evolution of Pointer Based Dynamics with Touchscreens

With the growth of touch sensitive screens and their application in consumer electronics, biometric authentication from touch screen dynamics is emerging. In touch dynamics, fingers or point devices are used to perform gestures, access control elements and interact with the user interface on the touch sensitive screen. The underlying granular data for mouse dynamics and touch dynamics is similar as both technologies rely upon gestures and pointer movements for authentication. Thus, the technology developed for MDA, with necessary modifications, can be used in touch dynamics.

2.2.8 Touch Screen Technology

Initially developed in the 1960s for air traffic control systems, touch screen are common today in devices such as ATMs, self-service kiosks at grocery stores, airports etc. However, the use of touch screens has become ubiquitous due to their usage in mobile devices and portable tablet computers. There are many technologies used to create touch screens. However, the most commonly used screens are capacitive or resistive based. One type of resistive display consists of two layers of glass or acrylic panels that are coated with indium tin oxide. This allows the layers to conduct electricity and also possess a degree of electrical resistance. These layers are placed together while being separated by invisible spacers. Other types of resistive displays also exist. However, the general principle on which resistive displays work remains the same: when pressure is applied

at a certain point on the display, a contact is created between the electrically conductive layers. A voltage is then applied across the Y direction on the screen. Since the touch creates a voltage divider at that point, an electrode along the X axis measures the voltage gradient, thus detecting the Y coordinate of the pressure point. Similarly, the voltage gradient across the X axis is measured to determine the Y coordinate of the point of contact. The coordinate sampling is performed multiple times and an averaging algorithm is used to determine the location of the contact [21]. The time to sense a single touch thus depends on the averaging algorithm, the analog technology used, and the clock speed of the device. Resistive screens thus work based on the pressure applied to compress the electrically conductive layers together. Due to the technology's passive nature, any pointing device (finger, stylus, etc) can be used to interact with resistive displays. They are also more economical than capacitive displays. However, resistive displays suffer from low responsiveness due to the need to apply pressure to create a contact.

Capacitive screens, although more expensive, have become prevalent in mobile devices such as cell phones and tablet computers due to their responsiveness. The capacitive screen uses an array of surface electrodes to detect a touch. When the user touches the screen, the body's capacitance causes the finger to act as the second electrode. The glass layer between the surface electrode and the finger acts as the dielectric. Based on the location of the surface electrode in the array, the coordinates of the contact point is calculated. Due to the nature of capacitive technology, any object that does not possess conductance cannot be used for interacting with such displays.

The primary objective of touch screen technology is to detect the location of the touch. It does not possess the ability to measure the degree of pressure applied. However, capacitive technology can use the number of electrodes that are activated as a relative measure of the pressure applied. Thus, pressure detection on touch screens is possible only when a deformable object such as a finger is used to touch the screen. Any object that does not possess this characteristic (e.g., a stylus) cannot provide meaningful pressure information.

2.2.9 Application to Continual User Authentication

As with mouse dynamics based authentication, research in touch-based authentication can be classified into two types: entry-point based authentication (static) and continual authentication. In static authentication, user identification is based on a pre-defined token such as a password, or personal identification numbers (PIN), etc. In this case, authentication assumes the form of keystroke dynamics, supplemented with pressure information from touch gestures. In graphical passwords, this could take the form of a pre-defined gesture, or connecting a grid of points on the touch screen in a single gesture. This authentication system, currently in use on smart phones, has been extended to include pressure data, illustrating the feasibility of a touch dynamics static authentication system [19]. However, it is important to note that in the majority of touch screen devices, the pressure is sensed indirectly and not based on a

dedicated pressure sensor. When additional pressure is exerted, the finger tip flattens slightly and increases the contact area. The increase in contact area is interpreted as an increase in pressure exerted.

Table 2 summarizes the experimental setup and results of four studies that have been performed on touch-based continual authentication.

Frank et al. [24] authenticated users by analyzing their regular device use patterns over time, through 34 different features extracted from touch strokes. This study achieved an equal error rate between 0%-4%. The authors also tested for inter-session authentication that showed that touch-based authentication can be used for long term authentication.

Li et al. [47] evaluated the performance of a live implementation of a smart phone-based touch authentication system. The touch data was collected in the background from 75 users who were asked to freely use the devices for a number of days. The collected data was used to create a SVM-based classifier that exhibited an equal error rate of 3%.

Feng et al. [23] used 53 touch and gesture features for classification. Additionally, they created a special digital sensor glove to achieve highly accurate continuous identification. The glove was used to capture 36 triaxial angular rate features when users performed touch activity. The glove data was collected for 11 subjects and the classifier trained using Random Forest, J48, and Bayes network algorithms. The authors achieved an accuracy of 2.15% FAR and 1.63% FRR when the digital glove was used. Without the glove, they reported an accuracy of 11.96% FAR and 8.53% FRR.

Recently, Serwadda et al. [67] performed a benchmark analysis and released the first public dataset in this field. The benchmark analysis compared ten classification algorithms on a locally collected dataset to determine the best performing ones. The best performing classifier on their dataset was Logistic Regression with an EER between 10.5%-17.2% depending on the type of strokes used to develop the model (horizontal or vertical) and the screen orientation (portrait or landscape).

As can be seen from the related work, the research in touch based continual authentication is in its infancy. The number of features varied from 10 to 53 across the different studies. One study does not disclose the types of features used [23]. Some studies leverage feature selection [24, 47] and others do not [23]. Similarly, feature normalization is performed only by one study [24]. Some studies use EER for benchmark comparison [24, 47, 67] while others only report error rates [23].

There are other significant factors that have currently not been researched: Touch based devices come with varying form factors. The display size, aspect ratio, screen orientation (landscape or portrait), the touch sampling rate of the touch screen, and the touch screen hardware are some factors whose effects on user behavior are not well understood. Modern touch screen - based devices are equipped with accelerometers and gyroscopes. If device orientation, user movement and device perturbation during interaction is found to be a significant source of variation in user behavior, it would be possible to use data from these sensors to augment touch-based authentication with the movement data. Recent

research fuses user movement and touch dynamics [10].

2.2.10 Contributions to touch dynamics

In this work, we demonstrate that the user's posture, the device size and the device manufacturer have a significant impact on the authentication performance of a touch-based authentication system. We show that the attributes used in current state-of-the-art touch-based authentication systems lead to a user model that is incapable of providing constant, reliable performance when any of the above-mentioned three factors are changed. This area of research in touch dynamics has previously remained unexplored. The results presented in this work are the first of its kind and significantly important in the development of robust touch-based authentication systems.

Table 2: A Summary of the current studies on touch-based continual authentication. A part of this table is summarized from a benchmark study [67]

Study	**# of Users**	**# of features**	**Features enumerated?**	**Feature selection performed?**	**Feature normalization?**	**Outlier removal?**	**Classification algorithm**	**FAR**	**FRR**
Frank et al. [24]	41	27	Y	Y	Y	Short strokes removed	SVM	0-4	0-4
							kNN	0-4	0-4
Li et al. [47]	75	10	Y	Y	N	N	SVM	3	3
Feng et al. [23]	40	53	N	N	N	N	J48	14	12
							Random Forest	7.5	8
							Bayes Net	11.96	8.53
Serwadda et al. [67]	41-106	28	Y	N	N	N	Logistic Regression	13.8	13.8
							SVM	15.4	15.4
							Random Forest	16.5	16.5
							Naive Bayes	20.5	20.5

Chapter 3
Keystroke Dynamics

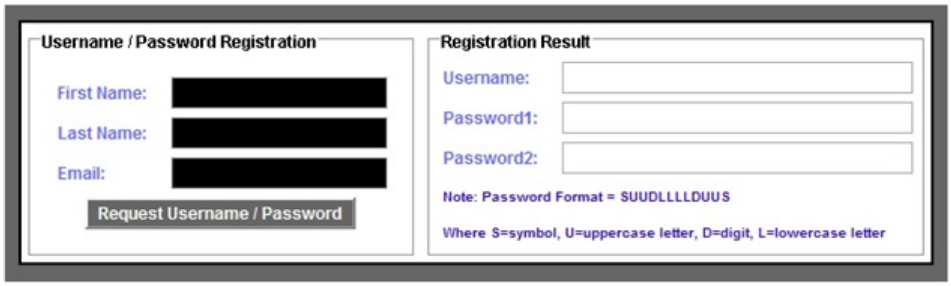
3.1 Datasets used

The experiments and analysis in Chapter 3 uses three keystroke dynamics based datasets. These were either collected locally or available publicly. They are labeled as KDS-1, KDS-2, and KDS-3. Each research question uses one or more datasets based on their suitability and availability. Thus, some research questions used only one dataset while othes used all three datasets. The collection procedure and characteristics of each dataset are described in the remainder of this section.

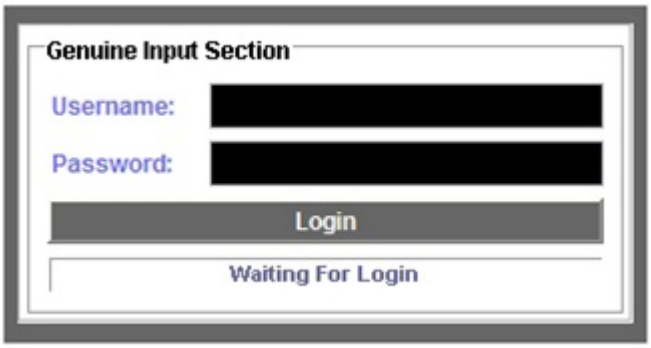
3.1.1 KDS-1 Dataset

The first data collection (KDS-1) was performed by Bartlow and Cukic [6]. They collected data from 41 users who typed short and long passwords repetitively. The characteristics and the content of KDS-1 is summarized in Table 3. The short passwords were 8-character lower case dictionary words, while the long passwords contained 12 characters in a uniform format **SUUDLLLLDUUS**, where **S** represents a symbol, **U** represents an uppercase character, **L** represents a lowercase character and **D** represents a numeric digit.

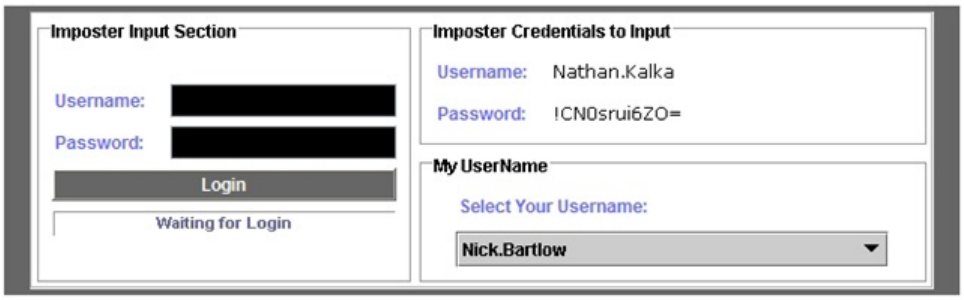
The data collection sequence consisted of enrollment and data entry. During enrollment, each user was given the short and long passwords through the interface shown in Fig. 1(a). Once the user was enrolled in the system, the data collection consisted of two distinct phases: training and testing. During the training phase, the user would enter their (genuine) username-password sequences regularly through the interface shown in Fig. 1(b). Following this entry, the user would be asked to enter the data that would correspond with the testing phase through the interface shown in Fig. 1(c). This would consist of providing another user's credentials to the current user. The user typed these credentials into the given interface. By doing so, the user acted as an impostor with respect to another user. Each user was asked to input the genuine and impostor data 10 times each in a day.



(a) Registration interface



(b) Genuine input interface



(c) Impostor input interface

Figure 1: User Interface for KDS-1 data collection

Table 3: Summary of KDS-1 and KDS-3 Datasets

	KDS-1 dataset	**KDS-3 dataset**
# of Subjects	41	51
Collection Format	Client/Server web based remote collection	In lab setting
Impostors present?	Yes	Yes
Who act as impostors?	Other users	Other users
Data collected	Password 1, Password 2	1 Password
Password format	8-letter, lower case dictionary word	".tie5Roanl"
	Format: SUUDLLLLDUUS	
Entries per user	40 entries per password	50 entries per session
User input device	Using personal computers	Lab PC
Frequency	5 times per day	1 session / sitting

3.1.2 KDS-2 Dataset

We collected this dataset for our research. In KDS-2, each user was assigned four passwords: 2 short and 2 long. The short and long passwords followed the same format as in KDS-1. We recruited 30 users for the study. They were coupled into 15 user pairs. Each user in a pair was assigned the same four passwords. The users were unaware of this pairing process. The pairing of users was done to simulate an environment where an impostor learns someone else's password, but does not have a chance to observe the genuine user's password typing rhythm.

The users were asked to submit their four passwords several times a day. The data was collected via a web-based front-end application built using Javascript and collected at a back-end server at our lab. The users used their own computers to submit data. The Javascript application collected and submitted the following information to the back-end server:

1. The keys pressed and the time stamps of the key events.
2. IP address.
3. Browser type.

4. Date and time of submission.

In this work, we only use key event data. The subjects were selected with reasonable variation in demographical characteristics so as to remove potential bias in experimental results. Each user was asked to submit at least 40 legitimate entries per password. The features of the dataset are summarized in Table 4. Most users submitted 50 - 55 entries per password. The minimum number of entries submitted for a password was 44. In order to create consistent experiments, as explained later, we used 40 entries per password from each user.

Table 4: Characteristics of KDS-2 Dataset

	Password Type	
	Short	*Long*
Min # of entries/user	103	90
Max # of entries/user	215	245
Avg # of entries/user	136	118
Total users	30	

3.1.3 KDS-3 Dataset

The KDS-3 dataset was collected by Killourhy and Maxion and is publicly available [46]. The dataset consists of 51 users typing the same password, `.tie5Roanl`, 400 times each. The passwords were entered by a user in 8 sessions. In each session, the user submitted 50 entries. Erroneous entries were removed and only legitimate entries using only the characters required were kept in the dataset. The characteristics of the dataset are summarized in Table 3.

Noise removal on KDS-1 and KDS-2 datasets

Both KDS-1 and KDS-2 were collected in an uncontrolled environment. This entailed that we analyze the dataset for outliers and remove them. The outliers were removed based on the total typing time distribution for a given user. For every user, those entries whose typing time was greater or less than $1.5 \times Inter-QuantileRange$ was removed from the dataset.

3.1.4 Advantages and Disadvantages of each KDS Dataset

In KDS-1 each user was assigned a short and a long password, impostors in the dataset represented all other users and would mimic a malicious user obtaining a password and attempting to circumvent an authentication scheme. Users were asked to submit data at their convenience with no specific requirements for frequency of submission per session. However, users were asked to submit no more than 5 entries per day.

In KDS-2 each user was assigned a pair of short and a pair of long passwords. Furthermore, each user was paired to another user. This creates the scenario where an impostor has gained access to a password and is habituating to it prior to launching an attack. Users were given the freedom to submit as many or as few entries per session and per day.

In KDS-3 all users were assigned the same password. This removes any confounding factors generated by using different passwords for each user. Such confounding factors related to location of letters in the password relative to the keyboard as well as the type of input device. However, the data collection procedure does not mimic real world situations as users will seldom submit a large number of password entries in a single session.

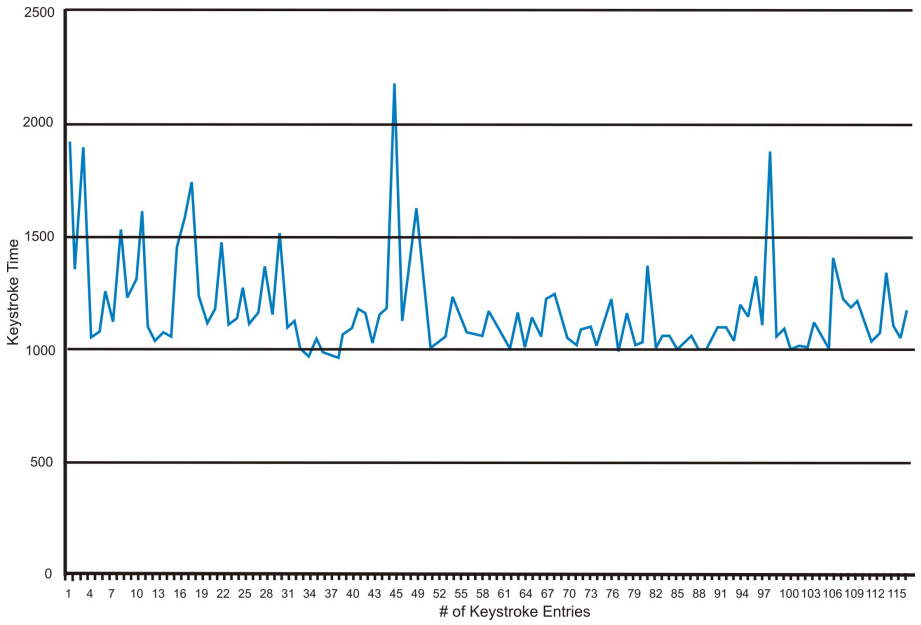
3.2 RQ1: When users are assigned unfamiliar passwords, is there a period of typing habituation?

In all three datasets, each long and short password pair was chosen by the system. Thus, all users were initially unfamiliar with their credentials. It is therefore expected that each user would have a learning phase before they acquire the ability to enter the credentials "smoothly".

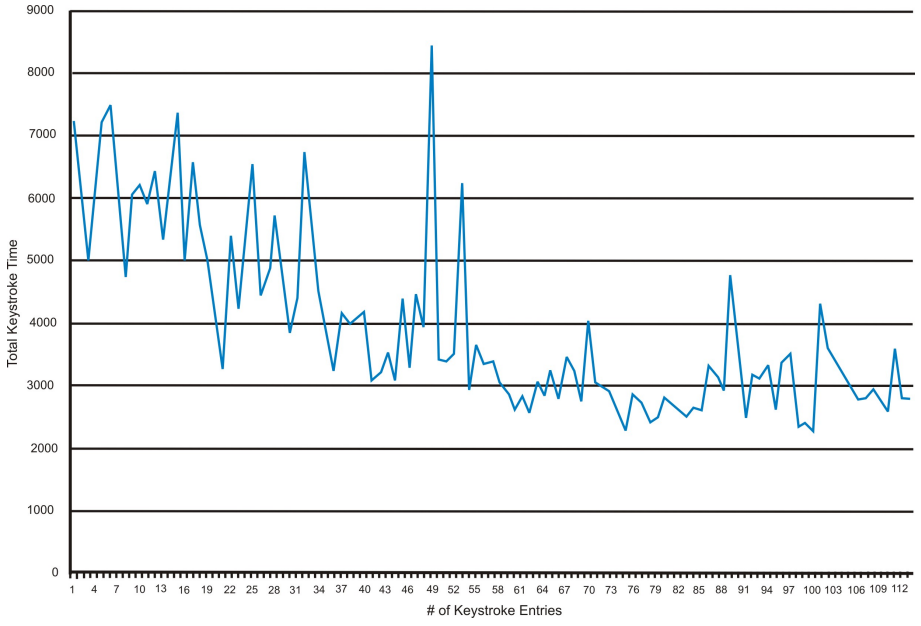
Fig. 2(a) shows the total time required by a typical user to type the short password over the course of the KDS-1 data collection. The difference between the maximum and minimum keystroke time over this period is 1.5 seconds. After excluding the major outliers, this difference falls to approximately 0.75 seconds. Over the course of the entire experiment, the total keystroke time for the short password does not appear to follow a significant habituation trend.

In order to further analyze any possible trend in the user habituation, we divided the short and long password typing entries of every user into groups of 5. Since the entries were arranged in a chronological order, the order of the entries intrinsically contained information on user habituation. Analyzing these groups of entries, or windows, provides information on user habituation.

In the second experiment, the short and long password entries were split into windows of 10 consecutive entries. Thus, two window sizes of 5 and 10 were used on both passwords. The variations in total typing time are plotted using box plots. A boxplot is a graphical representation of the five first order statistics - the minimum observation, lower quartile (Q1), median, upper quartile (Q3) and maximum observation. The box in the box plot is constructed using the inter-quartile range from the first to the third quartile. The solid horizontal line within the box represents the median value. The whiskers represent the smallest and the largest observations. The presence of outliers is noted with a hollow circle.

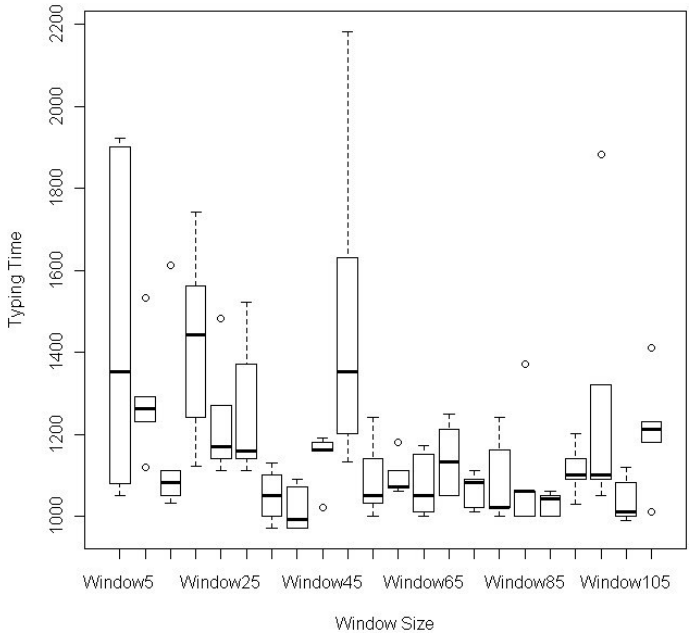


(a) Short passwords

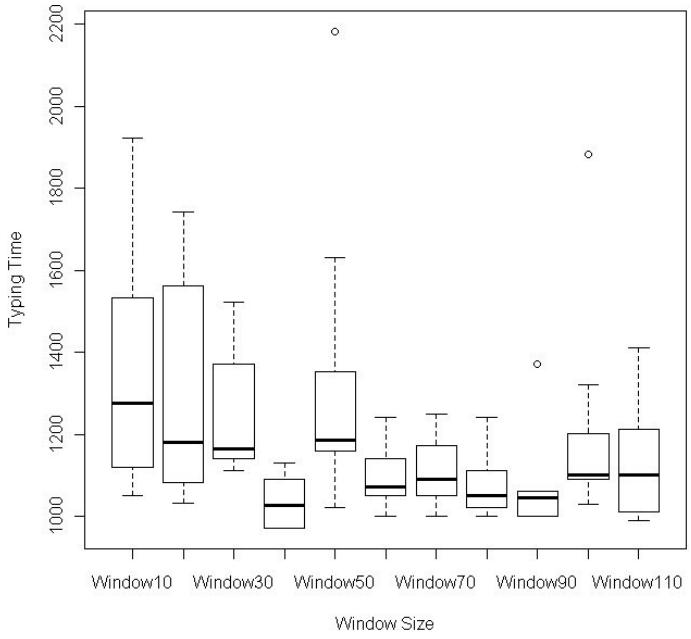


(b) Long passwords

Figure 2: Typical variation in total keystroke time for short and long passwords across the length of the study

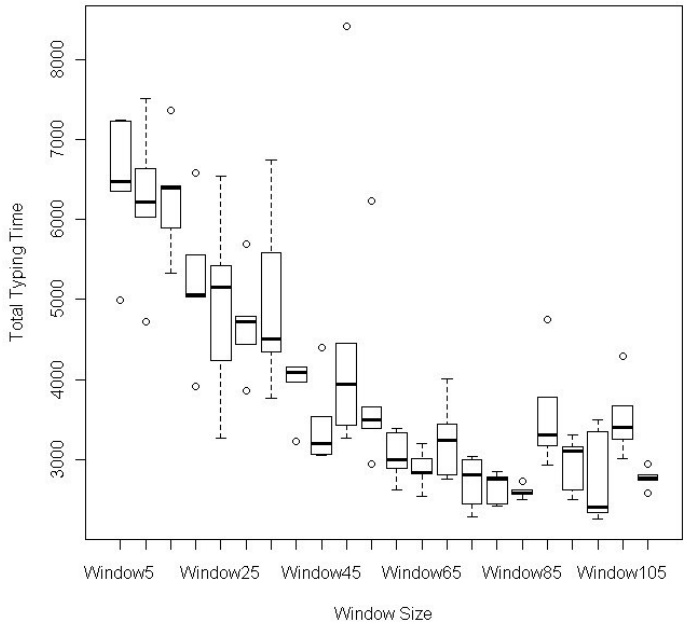


(a) Window size = 5 entries

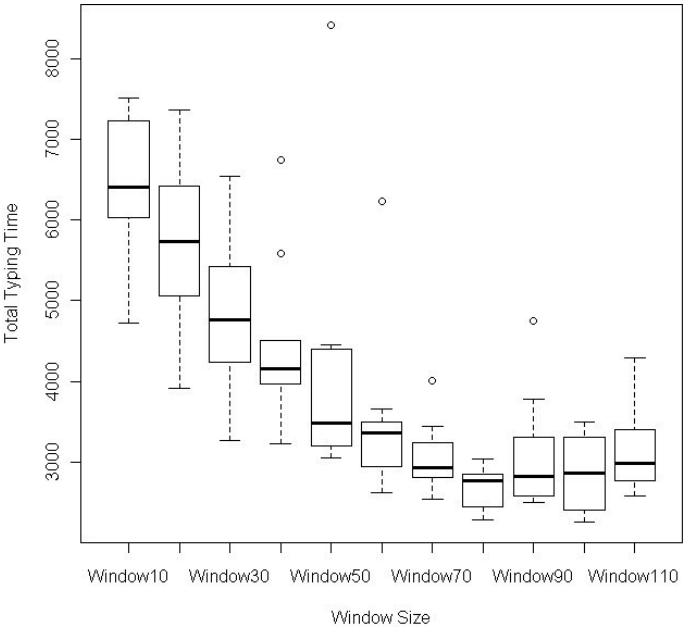


(b) Window size = 10 entries

Figure 3: Variation in keystroke time using different window sizes for a short password of a typical user



(a) Window size = 5 entries



(b) Window size = 10 entries

Figure 4: Variation in keystroke time using different window sizes for a long password of a typical user

3.2.1 Results and Analysis

Fig. 3(a) illustrates the variation in the typing pattern of a typical user for a *short* password over 5 consecutive entries. Fig. 3(b) depicts the same information for a window of size 10. Somewhat to our surprise, these two figures indicate that the total time of typing a *short* password does not exhibit a clearly observable reduction. Typing habituation for short passwords, if present, was minimal. A similar data pattern of behavior was observed for the short passwords of the remaining 31 users in the study.

Fig. 2(b) illustrates the total keystroke time for the *long*, complex passwords, for a typical user. The largest difference between the keystroke times was found to be 7.5 seconds. After excluding the major outliers, this difference was approximately 4.5 seconds. The results indicate a much clearer habituation trend than in case of short passwords. The decreasing trend of time needed to enter a long password is related to time between two consecutive password entries. When the user enters the long password after a significant time elapsed after the previous trial (day or days), it takes a few entries before the user settles into a rhythm. A similar behavior was noted for the remaining 31 users, which cannot be presented due to space restrictions.

Fig. 4(a) and 4(b) show the box plots for the total keystroke time for long passwords using a time window of 5 and 10 consecutive entries, respectively. The figures indicate that the mean time to type in the password is follows a decreasing trend over the course of the experiment. Furthermore, the typing pattern tends to become smoother, as evidenced by the fewer outliers and reduced variance. This behavior indicates that habituation is present in typing of the password. However, for complex passwords, users requires a significant number of entries prior to developing a uniform typing pattern. The behavior of this user was found to be similar for most of the remaining 31 users.

We can summarize our observations as follows:

1. User habituation appears to be minimal for short passwords. Short passwords are commonly occurring English words, familiar to users, such as *explorer*, *zeppelin*, etc. The average user may type such words during their daily activities and hence they exhibit minimal habituation patterns. The most common words (such as *computer*, *password*) showed the lowest habituation trends indicating the weakness of simple words as a password mechanism in keystroke based user authentication.

2. For passwords randomly generated passwords, there is a decrease of 46% in total typing time between the first 20 entries and last 20 entries. Each password was comprised of special characters, numerals, and upper and lower case characters, such as *[LO2uqam8UI+* , *;PG3xuel9LU}*, etc. The average user is not familiar with such random sequences and exhibited a marked habituation trend as he or she learned the words and improved typing over time.

3. Using too small a window size for habituation analysis can cause the graph to be too granular. Keystroke patterns do not follow a perfectly smooth

trend. Rather, the keystroke entry times fluctuate locally as the user habituates, while the local mean decreases with time. Using a correct window size was necessary to remove the effect of local variations in the data.

3.3 RQ2: Does user habituation play a critical role in building effective keystroke dynamics classification systems?

How can user's password typing habituation benefit authentication? When keystroke dynamics is used to authenticate a user, in addition to the knowledge of the password itself, more stable typing patterns should enable more precise methods for authentication. In this section, we explore the effect of user habituation on the performance of user classification, a mechanism that utilizes keystroke dynamics in authentication. Using the same dataset, we will analyze if models developed from longer sequences of consecutive password typing attempts outperform those that use shorter training sequences.

A classifier is built to separate genuine password entries from unauthorized (impostor) attempts. We trained it using genuine and impostor typing entries (the dataset includes keystrokes of subjects typing their own as well as each other's passwords). We built user typing classification models by steadily increasing the size of training entries from 6 to 30 attempts of typing the same passwords, in steps of 3. The test samples for genuine and impostor typing attempts were those recorded after the first 30 in their respective category. Using this system, the average test size was 58 and 59 sequences for short and long passwords respectively. The experiments were carried out using the Random Forest classifier [13]. We calculated the Equal Error Rate (EER) for each user (30 users for long passwords and 33 users for short), and then calculated the average EER for differing number of samples in the training sets. The average EERs across all users as the training set size is varied for short and long passwords are listed in Table 5 and plotted in Fig. 5. The results can be summarized as follows:

1. As the training set was incrementally increased, the average EER decreases. The maximum value of errors for short passwords is 14.9% when 9 samples are used in the training set, while a 7.0% EER is achieved as the number of training samples is increased to 30.

2. For long password, the EER decreases from 12.7% to 3.7%. This observation is further strengthened by Fig. 6 that depicts the ROC curves for long password using different number of training samples. As higher the number of training samples, the better the authentication performance.

3. On an average, using keystroke dynamics with long passwords resulted in 3.7% fewer authentication errors compared to short passwords.

4. It is important to note that in the KDS-1 dataset, impostors were given the passwords of their peers. Without keystroke dynamics, all attempts to authenticate would succeed. With keystroke dynamics, about 90% to 96.4% of them would be successfully detected and disallowed.

Table 5: Change in EER when the size of the training set is increased for short and long password schemes.

Number of training vectors	**Average EER**	
	Short	*Long*
6	13.5%	12.7%
9	14.8%	8.7%
12	12.1%	7.9%
15	10.2%	6.7%
18	9.2%	5.6%
21	8.6%	5.2%
24	8.5%	4.2%
27	7.4%	3.7%
30	7.0%	3.7%
Average	10.2%	6.5%

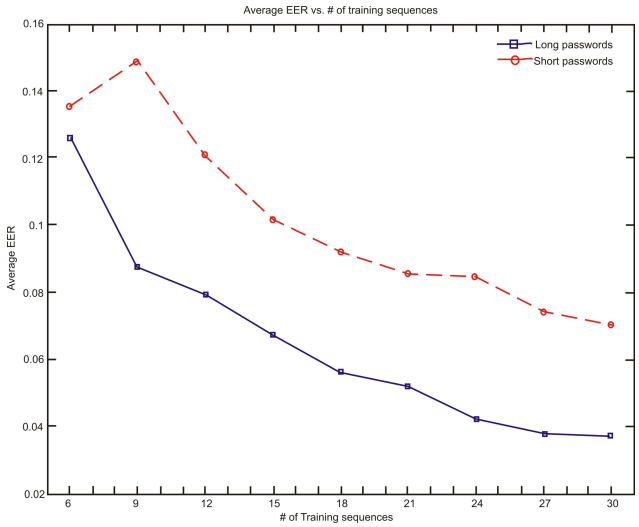


Figure 5: A graphical illustration of change in EER as the training set size is increased

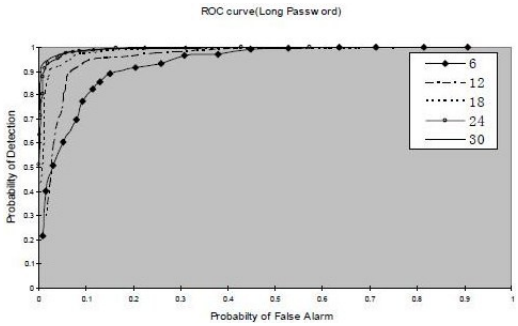
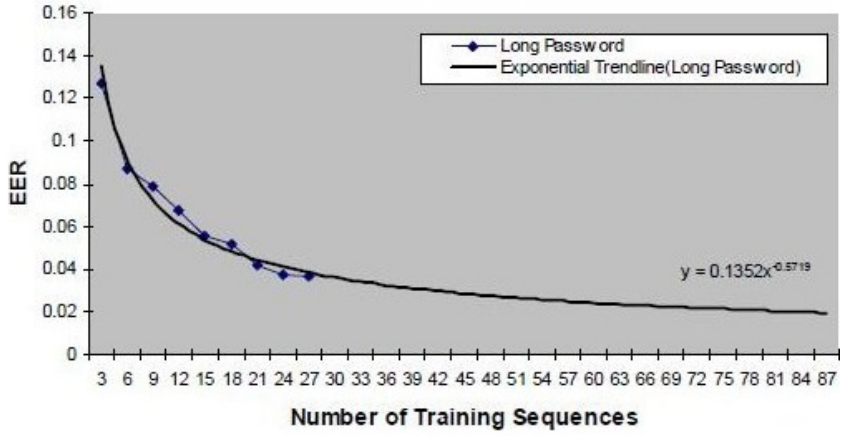


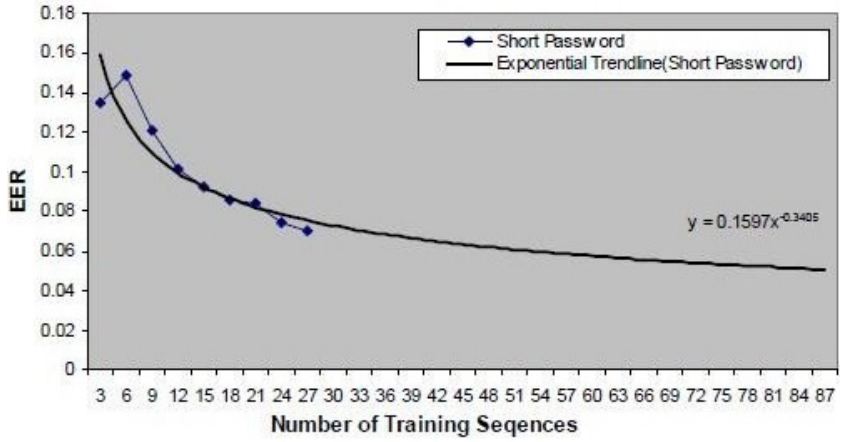
Figure 6: ROC for long passwords as the size of the training set is varied between 6-30 samples

This performance increase is a natural consequence of using more data to train the classifier. However, user habituation also plays a role in this performance increase. An absence of user habituation would have resulted in a gentler decline in the EER and could have also caused a significant deviation from the smooth decline observed in Fig. 5. Due to the nature of the data and the characteristics of the classifier, studying user habituation exclusively without the effect of confounding factors is difficult. User habituation can only be evaluated by incrementally collection of typing samples. However, the classifier improves due to the increase in the number of training samples too. Due to this relationship between the user habituation and classifier algorithm, it is difficult to study the impact of saturation effect in user habituation in isolation.

To forecast the performance of keystroke - based authentication, we used trend lines to determine the EER related to the number of samples used for training. Fig. 7(a) and 7(b) illustrate the trendlines for the long and short passwords, respectively. The trend line for long passwords function is $y = 0.1325x^{-0.5719}$, and $y = 0.1597x^{-0.3405}$ is for short passwords, with y being the EER and x being the number of training samples. Based on our data analysis, the performance of authentication would increase by less than $1\,\%$ if more than 30 typing samples are used in training. While this number may vary based on the user demographics and experimental environment, it provides a reasonable baseline for future measurements. It is also important to note that once the user is habituated to the password, rebuilding the classifier would result in better performance. This is because data from earlier typing samples no longer represent the user as he / she habituated to the complex password.



(a) Long passwords



(b) Short passwords

Figure 7: EER Projections for a typical password based classifier

3.4 RQ3: Does habituation have a statistically significant effect on the user's Total Typing Time?

3.4.1 Experiment Setup

In order to statistically analyze the effect of habituation in Sections 3.4-3.7, we accumulate consecutive password entries from each user in the dataset under analysis into windows. Fig. 8 illustrates the concept. The sample keystroke entry vector is converted to windows of size 3 by computing the median value for each feature. The size of the window refers to the number of entries used to compute it. Using a large window size smoothes out irregularities in typing rhythm caused by external factors or noise (physical or mental state, change in posture, etc.). However, too long a window size may obscure information about changes in typing over time. Similarly, a small window size shows variations between keystroke entries.

After evaluating various window lengths, between 1 and 10, we chose a default window length of 10. Authentication classifiers performed best when using 10 entry windows. Longer windows were not practical because of two reasons: a real-world authentication system should not require excessive amount of data for training. We also needed at least 4 windows to perform analysis. Having collected 40 entries from every user, we split the dataset into 4 windows. Thus, *Window* 1 will contain data from keystroke entries $1 - 10$, *Window* 2 contains entries $11 - 20$ and so on.

In order to assess the statistical significance of our experiment, we employed a non-parametric test, the Friedman's Test [25]. It is used to test for differences between three or more paired groups when the dependent variable being measured is ordinal. The groups in this experiment are the windows. The N subjects and k windows are considered separate independent variables in the analysis. The test statistic for the Friedman's test is a Chi-square with $(k - 1)$ degrees of freedom. Thus, the hypotheses for comparison across the windows are:

H0: The distributions are the same across the windows

H1: The distributions across the windows are different

Sample Keystroke Entry Vectors

1a

User 1 - Short Pwd 1	Key Down - Key Down Times (msecs)								Key Hold Times (msecs)								Key Delay Times (msecs)								Total Time (msecs)
	w-h	*h-e*	*e-e*	*e-l*	*l-i*	*i-n*	*n-g*	*g-NULL*	*w*	*h*	*e*	*e*	*l*	*i*	*n*	*g*	*w-h*	*h-e*	*e-e*	*e-l*	*l-i*	*i-n*	*n-g*	*g-NULL*	
Entry 1	83	149	172	93	182	220	140	0	104	84	95	109	76	75	85	110	-21	65	77	-16	106	145	55	0	1149
Entry 2	150	1031	180	145	200	210	80	0	90	85	100	105	90	141	85	100	60	946	80	40	110	69	-5	0	2096
Entry 3	99	141	160	90	185	216	102	0	104	85	89	105	85	85	87	100	-5	56	71	-15	100	131	15	0	1093
Entry 4	110	155	509	130	191	190	152	0	120	105	110	100	85	75	80	93	-10	50	399	30	106	115	72	0	1530
Entry 5	101	96	181	108	166	190	95	0	112	86	105	109	82	85	101	95	-11	10	76	-1	84	105	-6	0	1032
Entry 6	96	124	136	111	215	210	96	0	102	75	70	106	85	80	91	85	-6	49	66	5	130	130	5	0	1073

Sample Keystroke Entry Windows

1b

User 1 - Short Pwd 1	Key Down - Key Down Times (msecs)								Key Hold Times (msecs)								Key Delay Times (msecs)								Total Time (msecs)
	w-h	*h-e*	*e-e*	*e-l*	*l-i*	*i-n*	*n-g*	*g-NULL*	*w*	*h*	*e*	*e*	*l*	*i*	*n*	*g*	*w-h*	*h-e*	*e-e*	*e-l*	*l-i*	*i-n*	*n-g*	*g-NULL*	
Window 1	99	149	172	93	185	216	102	0	104	85	95	105	85	85	85	100	-5	65	77	-15	106	131	15	0	1149
Window 2	101	124	181	111	191	190	96	0	112	86	105	106	85	80	91	93	-10	49	76	5	106	115	5	0	1073

Figure 8: Short password feature vector and window example. Fig. 8(a) illustrates six consecutive entries from User 1. Fig. 8(b) summarizes these entries using a window of size 3 and taking the median of Entries $1-3$ to create Window 1 and so forth

Here, 'distribution' refers to the distribution of typing times in any given window. For a set of N subjects, 3 groups (windows) are being compared to test if there are differences in typing times between any two. The values (typing times) across each row are rank-ordered. The resulting ranks are then summed for each column. The null hypothesis asserts that there is no difference amongst the 3 groups. Therefore, the sum of ranked scores in each column R_j should approximately be $\frac{k(N+1)}{2}$. To measure the degree to which each observed sum of ranked column varies from the null hypothesis value, the Chi-square statistic χ_r^2 for Friedman test is calculated as

$$\chi_r^2 = \frac{12}{Nk(k+1)} \sum_{j=1}^{k} R_j^2 - 3N(k+1)$$

The results of the Friedman test are defined by the parameters χ_r^2 and p. The test statistic χ_r^2 is distributed according to the χ_r^2 distribution with $k-1$ degrees of freedom. The value p defines the probability that the calculated χ_r^2 value will be obtained if the null hypothesis were true. In our experiment, we chose $\alpha = 0.05$ as the significance threshold. Thus, a p-value under 0.05 indicates the rejection of the NULL hypothesis at the 5% significance level.

The Friedman's test only provides a test to determine if there is a difference in the distribution of typing times amongst the windows. It does not provide any information on which pair(s) of windows are different. For this information, a post-hoc analysis is carried out using Wilcoxon signed-rank tests for multiple comparisons between the treatments. A Bonferroni correction is applied to adjust for the multiple comparisons that are being made [11].

In order to measure if habituation causes a statistically significant reduction in the user's total typing time, user entries were split into windows of size 10. For each window we calculated the median total typing time for 10 consecutive entries. For all three datasets (KDS-1, KDS-2 and KDS-3) we utilized 4 windows of observation.

Fig. 9 illustrates the structure of the Friedman's test for our setup. If the difference in median typing time between the two windows is statistically significant for all users, this implies that the users' typing rhythms changed significantly between the two windows. Note that, contrary to the example in Fig. 9, the two windows need not be consecutive. Skipping a window can accommodate the analysis of individuals whose password typing habituation is slower. While this test does not measure the change in typing patterns for each individual keystroke in the password, it does provide a general means to assess habituation.

3.4.2 Hypothesis Test

We defined the hypotheses for this test as follows:

NULL: There is no statistically significant reduction in the median typing time between the two windows of observation.

User	Window 1 Median Total Typing Time Across 10 entries	Window 2 Median Total Typing Time Across 10 entries	Window 1 - Window 2
User 1 - LongPwd 1	10729.5	8671	2058.5
User 1 - LongPwd 2	11446.5	9226.5	2220
User 2 - LongPwd 1	8387.5	5886.5	2501
User 2 - LongPwd 2		4004	2327
...	...	...	...
User 25 - LongPwd 1	6573.5	5607.5	966
User 25 - LongPwd 2	7355	5672.5	1682.5
User 26 - LongPwd 1	6759	6344	415
User 26 - LongPwd 2	5871	6075.5	-204.5

Figure 9: Long password median total time values for window size 10. Window 1 refers to the median typing time for entries 1 through 10 for each user. Window 2 refers to the median typing time for entries 11 through 20 for each user. The Window 1 - Window 2 column depicts differences in median typing time.

ALT: There is a statistically significant reduction in median typing time between the two windows of observation.

We tested for change in typing time between all possible combinations of window pairs (W_{1-2}, W_{1-3}, W_{1-4}, W_{2-3} and W_{3-4}) for both datasets. Datasets KDS-1 and KDS-2, as explained earlier, contain short and long passwords and the tests were carried out on both types of passwords.

3.4.3 Short Passwords

Table 6: Statistical significance of reduction in t_{total} for short passwords (KDS-1, KDS-2) at $\alpha = 0.05$.

Friedman's test		*Post-hoc analysis*				
		W_{1-2}	W_{1-3}	W_{1-4}	W_{2-3}	W_{3-4}
KDS-1 Short *p=0.0056*	p	0.002	0.0001	0.0097	0.0554	0.1321
	Sig.?	YES	YES	YES	NO	NO
KDS-2 Short *p=0.0019*	p	0.0067	0.0008	< 0.0001	0.0412	0.007
	Sig.?	YES	YES	YES	YES	YES

Table 6 summarizes the results of Friedman's Test on short passwords from the KDS-1 dataset. We infer the following:

- When compared to Window 1 (W_1), all subsequent windows consistently show a statistically significant decrease in typing time. This shows that habituation, expressed as the decrease in a user's typing time due to the learning the password over time, does occur for short passwords.

- The window pairs W_{2-3} and W_{3-4} do not show a statistically significant decrease in typing time. This indicates that, at least in KDS-1 dataset, the users habituated to the password within the first 20 or so attempts.

Table 6 also summarizes the results of our statistical study utilizing the short passwords from KDS-2 dataset. From these results, we draw the following observations:

- Just as in KDS-1, when compared to W_1, all subsequent windows show a statistically significant decrease in typing time (at $\alpha = 0.05$), thus reinforcing the conclusion that habituation does occur for short passwords.
- Unlike KDS-1, the decrease in typing time remained statistically significant for window pairs W_{2-3} and W_{3-4}. Thus, the habituation process for a user continued through all 40 entries. We attribute this result to a slight variation in data collection protocols, in KDS-2 each user had to memorize two short and two long passwords.

The next question is whether the observed difference in typing time can be effectively leveraged to create better authentication classifiers based on keystroke dynamics. This question is explored later in Section 3.7.

3.4.4 Long Passwords

Tables 7 summarizes the results of our statistical analysis on the long password data from KDS-1, KDS-2 and KDS-3, respectively.

Table 7: Statistical significance of reduction in t_{total} for long passwords from KDS-1, KDS-2 and KDS-3 at $\alpha = 0.05$

Friedman's test		*Post-hoc analysis*				
		W_{1-2}	W_{1-3}	W_{1-4}	W_{2-3}	W_{3-4}
KDS-1	p	< 0.0001	< 0.0001	< 0.0001	0.020	0.0004
p<0.0001	Sig.	YES	YES	YES	YES	YES
KDS-2	p	< 0.0001	< 0.0001	< 0.0001	0.0095	0.0001
p<0.0001	Sig.	YES	YES	YES	YES	YES
KDS-3	p	< 0.0001	< 0.0001	< 0.0001	0.0042	0.0935
p<0.0001	Sig.	YES	YES	YES	YES	NO

The observations drawn from the results in Table 7 follow:

- In all three datasets, when compared to W_1, all windows show a statistically significant decrease in total typing time at $\alpha = 0.05$ indicating the effects of habituation.
- In datasets KDS-1 and KDS-2, users continue to show a significant reduction in total typing time even after typing the password 40 times. However, in KDS-3 users did not show a statistically significant reduction in typing

time after 30 keystroke entries. There are two evident reasons for this: the two long passwords used in KDS-1 and KDS-2 are longer and more complex than the one password used in KDS-3. Also, the subjects in KDS-1 and KDS-2 were not allowed to type more than 5 times a day to simulate a realistic scenario, whereas the subjects in KDS-3 typed one password 50 times in a single session. Due to the increased password complexity and reduced frequency of typing it, subjects in the first two datasets exhibited a more protracted habituation period. It is natural for users to habituate faster when they enter the same password in quick succession.

- In KDS-1 and KDS-2, the strength of rejection of the NULL hypothesis is much higher than that for short passwords. In fact, in the majority of the window pairs, the p value is smaller by an order of magnitude as compared to the p values for the short password set. This indicates that the decrease in typing time is larger and, thus, the habituation more pronounced for long passwords.

- In KDS-1 and KDS-2, the average reduction in total typing time was between 1.5 and 3 seconds for the long password in both datasets. Such a reduction reflects the complexity of the passwords. As for the short passwords, the rejection of the NULL hypothesis is again stronger in dataset KDS-2, in which users were asked to memorize twice the number of credentials as compared to KDS-1.

- The graph in Fig. 10 plots the change in Total Typing Time for a typical user in the KDS-3 dataset. Each datapoint refers to the mean Total Typing time for a given Session / Window combination. The datapoints are grouped for the same Window in different sessions. The graph shows that the difference in typing time between the first and last window progressively diminishes according to the number of sessions that the user undertakes. For the first two sessions, each Window in a session exhibits a lower mean typing time as compared to the previous window. This phenomenon is not observed after the first 2 sessions, i.e. first 100 entries.

3.4.5 Conclusions

The analysis of all three datasets reveals that user habituation, measured through the the reduction in the total typing time (or increase in typing speed) needed to complete a password is evident in both short and long passwords. However, the effects of habituation were found to be dependent on the password type as well as the data collection process. In KDS-1, users were asked to memorize one short and one long password. As a result, users appear to have exhibited quicker memorization of the passwords. In contrast, users in KDS-2 were asked to memorize four passwords. As a result, it appears that it took longer to memorize and type the passwords.

In KDS-3 the users were assigned a single password and asked to submit 50 successive entries in each session. As a result, users did not exhibit a marked

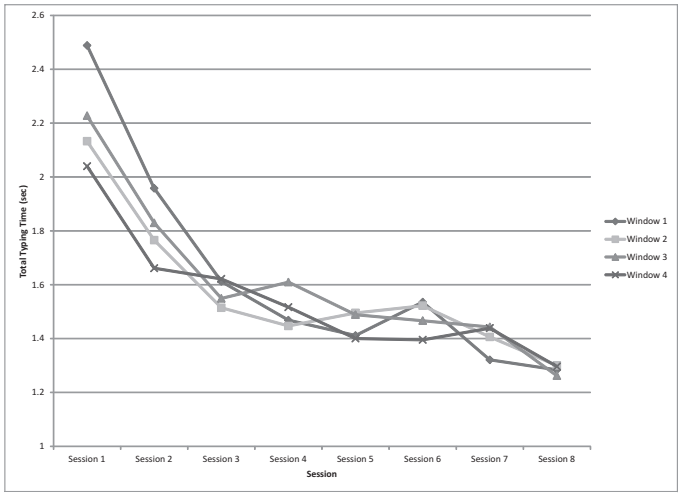


Figure 10: The change in mean Total Typing Time for each Window across the 8 Sessions for a typical user in KDS-3 Dataset

habituation phase when comparing windows after the first 30 entries. In a real world environment users will seldom be asked to submit 50 entries. In contrast, KDS-1 and KDS-2 allowed users to submit credentials at their convenience over a $4-5$ week period of time. However, KDS-3 was collected in a controlled environment that eliminates the effect of confounding factors such as user environment, keyboard used, etc.

Overall, habituation is more pronounced in the long passwords with a typical reduction in typing time between 1.5 and 3 seconds. While the NULL hypothesis is rejected for both the short and long passwords, the rejection is stronger in the latter case. This can be attributed to the length and complexity of the long passwords which makes them more challenging to memorize and retype.

3.5 RQ4: Does habituation have a statistically significant effect on the variance of the user's Total Typing Time?

The results of the above experiment show that users' total typing time of passwords decreases significantly over time. We next theorized that habituation could also be described by the reduction in variance in typing time across successive windows of observation. To further study the effect of new password adoption, we decided to test if it causes a reduction in the variance of the total typing time just as it reduced the total typing time. The following sub-sections omit the analysis results of KDS-1 as its results agreed with those obtained from KDS-2.

Table 8: Statistical significance of the reduction in variance of t_{total} for passwords in KDS-2 and KDS-3 at $\alpha = 0.05$

Friedman's test		*Post-hoc analysis*				
		W_{1-2}	W_{1-3}	W_{1-4}	W_{2-3}	W_{3-4}
KDS-2 Short p=0.028	p	0.0369	0.003	0.088	0.157	0.0787
	Sig.	YES	YES	YES	NO	NO
KDS-2 Long p=0.007	p	0.0008	0.005	0.0004	0.4621	0.2382
	Sig.	YES	YES	YES	NO	NO
KDS-3 p<0.004	p	< 0.0001	< 0.0001	< 0.0001	0.3574	0.1915
	Sig.	YES	YES	YES	NO	NO

We now compare median variance between pairs of windows for each user. The variance in t_{total} is computed for each window of 10 entries and then compared to other windows using the Friedman's Test. The following hypotheses are tested:

NULL: The variance in total typing time between the windows of observation is not significantly different.

ALT: The variance in total typing time between the windows of observation is significantly reduced over time.

3.5.1 Short Passwords

Table 8 demonstrates the statistically significant reduction in variance of total typing time for short passwords in KDS-2. From these results we observe that, when compared to W_1, all windows show a statistically significant decrease in variance at $\alpha = 0.05$, indicating the effect of habituation.

3.5.2 Long Passwords

Table 8 demonstrates statistically significant reduction in variance of total typing time for long passwords in KDS-2. From these results we observe that, when compared to W_1, all windows show a statistically significant decrease in variance at $\alpha = 0.05$, indicating a pronounced habituation.

3.5.3 Analysis of KDS-3 Dataset

Table 8 demonstrates statistically significant reduction in variance of total typing time for long passwords in KDS-3. From these results we observe that, when compared to W_1, all windows show a statistically significant decrease in variance at $\alpha = 0.05$, indicating a pronounced habituation. It is interesting to note that the NULL hypothesis is rejected more strongly as we compare later windows to

User 1 (U1)	Key Down - Key Down Times (msecs)								Hold Times (msecs)								Delay Times (msecs)								Total Time (msecs)
	f-o	o-u	u-n	n-t	t-a	a-i	i-n	n-NULL	f	o	u	n	t	a	i	n	f-o	o-u	u-n	n-t	t-a	a-i	i-n	n-NULL	
Window 1 (W1)	113	65	152	75	194	71	79	0	116	100	104	113	77	146	107	98	4	-35	61	-35	115	-75	-28	0	859
Window 1 (W2)	146	30	171	106	169	70	74	0	142	71	70	136	76	179	117	103	4	-36	71	-36	93	-80	-64	0	831

User 1 (U2)	Key Down - Key Down Times (msecs)								Hold Times (msecs)								Delay Times (msecs)								Total Time (msecs)
	f-o	o-u	u-n	n-t	t-a	a-i	i-n	n-NULL	f	o	u	n	t	a	i	n	f-o	o-u	u-n	n-t	t-a	a-i	i-n	n-NULL	
Window 1 (W1)	151	253	248	143	648	185	253	0	104	108	105	105	105	133	106	105	42	146	170	38	543	44	170	0	1982
Window 1 (W2)	126	233	241	119	323	159	237	0	108	104	101	109	105	107	103	75	22	129	138	31	219	56	145	0	1690

Diff (U1W1 - U2W1)	-38	-188	-96	-68	-454	-114	-174	0	12	-8	-1	8	-28	13	1	-7	-38	-181	-109	-73	-428	-119	-198	0	-1123
Diff (U1W2 - U2W2)	20	-203	-70	-13	-154	-89	-163	0	34	-33	-31	27	-29	72	14	28	-18	-165	-67	-67	-126	-136	-209	0	-859

Euclidean Distances	W1	W2	W3	W4	W5	W6	W7	W8
Between U1 and U2	1360	984	618	905	848	797	873	714.728

Figure 11: Euclidean Distances between a pair of users for the short password `fountain`. Each window is comprised of the median typing times for 5 consecutive entries. The Diff values indicate the differences in each feature. The Euclidean distances are computed as the square root of the sum of squares of Diff scores.

W_1. This is once more due to the fact that the users are likely to show higher variance in total typing time during the earlier part of each session.

3.5.4 Conclusions

For both short and long passwords, the reduction in variance in total typing time was significant for all data windows when compared to Window 1. Windows 3 and 4 did not show a significant reduction in typing variance with respect to Window 2, indicating that the variance of password typing period diminishes with time, consistent with expectations. Therefore, based on variance alone, we speculate that habituation to a new password tends to plateau after time.

3.6 RQ5: Does user separability change with time?

In Sections 3.4 and 3.5, we showed that the total typing time and its variance decrease with time. To study the effects of habituation further, we investigated the change in separability between user pairs who are assigned the same password.

We measure the separability using the Euclidean distance between all windows, for every user pair. The windows, same as before, consist of feature vectors representing n consecutive keystroke entries. The Euclidean distance between windows is defined as the distance between the median feature vectors of those windows. A window size of 5 is selected for this study. A window size of 5 is chosen to allow a higher level of granularity to determine how separability changes over time. Fig. 11 illustrates Windows 1 and 2 for a pair of users who were both assigned the same short password (`fountain`), as well as the Euclidean distance of difference vectors across the two windows.

Table 9: Statistical significance of user separability, measured through Euclidean distance after the user submitted 5, 20, and 40 password entries; $\alpha = 0.05$

Friedman's test	*Post-hoc analysis*				
p=0.009		$W_1 - W_4$		$W_1 - W_8$	
		Short	Long	Short	Long
	p	0.31	0.005	0.096	0.013
	Sig.	NO	YES	NO	YES

In order to determine whether there are changes in separability between users forming a pair, we use Friedman's Test to compare the Euclidean distances between 3 windows of observation: W_1, W_4, and W_8. The Friedman's test is a non-parametric test that is used for repeated measures over the same users over time. The following hypotheses are tested for both short and long passwords:

NULL: Euclidean distance between user pairs does not change over time.

ALT: Euclidean distance between user pairs changes over time.

As part of the the test's methodology, the statistical comparison was carried out on $W_1 - W_4$, $W_4 - W_8$ and $W_1 - W_8$. However. the results for $W_4 - W_8$ were discarded as we were not interested in them. The results of this experiment are shown in Table 9 and can be summarized as follows:

- Separability between users typing the same short password does not change significantly with time. In other words, the inter-user distance remains relatively constant.
- The inter-user separability changes significantly for long passwords. Interestingly, the absolute values within the Euclidean distance vector between the users decrease with time. It appears that the user typing patterns become increasingly similar. This result is somewhat counterintuitive, as one would expect that given the complexity of the long passwords, users would be easier to differentiate.

The reason for a decrease in the relative distance between the users who type the same long password is simply a consequence of the learning process. In early entries, users demonstrate high variability in keystroke patterns and the relative distance between users is large. Over time as the users accommodate to their new passwords, the variability of their keystroke entries is reduced, the keystroke patterns become more uniform. Therefore, the distance between paired median values of feature vectors decreases without ever reaching 0. Although inter-user distances decrease intra-user variance also decreases, thus enabling us to develop keystroke authentication classifiers whose performance improves through habituation. We show this result later in Section 3.7.3.

3.7 RQ6: What is the effect of habituation on classifier performance?

3.7.1 Selecting the Best Distance Measure

In order to perform authentication, we need to be able to separate user's password typing patterns. One way to do this is through vector distance measures. We tested the following distance based classification approaches:

1. Manhattan Distance (scaled)
2. Outlier Count (z-score)
    (a) Mahalanobis distance based measures:
    (b) Mahalanobis (nearest neighbor)
    (c) Mahalanobis (normed)
    (d) Mahalanobis

These algorithms were selected based on the results of Killourhy and Maxion [46]. They tested 14 distance measures from keystroke-dynamics literature and ranked their performance. In that study, the distance measures we list above performed the best. Distance measures with a threshold can serve as one-class classifiers. Threshold set-up requires training to be able to differentiate between nominal and abnormal patterns. In KDS-2, we collected data from 30 users. We created classifiers using Window 3 of each user for training. The classifiers were tested on Window 4. KDS-2 offers paired users, i.e., each password is given to two individuals unbeknown to them. This allows us to test the typing pattern of user A against her own samples as well as against the typing samples of A_{pair}, the impostor who knows the password. Windows 3 and 4 were chosen for training and testing to minimize the effect of variation that occurs in the early keystroke entries. Password typing entries are feature vectors and their content has been described in Section 3.1.2.

The distance measure computes a score for each genuine and impostor entry in their respective testing windows ($W4_A$ of User A and $W4_{A_{pair}}$ for User User A_{pair}). The manner in which each algorithm trains and then computes the user and impostor scores have been discussed extensively in literature. [46] explains the procedure to compute user and impostor scores for this dataset.

Generated scores are used to compute the ROC curve of each classifier. The ROC curve provides a graphical representation of the classifier's performance. It plots a relationship between the classifier's False Accept Rate (FAR) and the False Reject Rate (FRR). The FAR refers to the frequency with which the classifier misses the detection of impostors. The FRR refers to the frequency at which a genuine user is rejected as an impostor. The zero-miss False Reject Rate (ZM-FRR) and Equal Error Rate (EER) are two common measures used to compare the performance of different classifiers without plotting the ROC curve. The EER is calculated by adjusting the classifier threshold such that the

Table 10: Performance metrics for the best classifiers from [46] on the password entries from KDS-2 and KDS-3. For KDS-3, data from Session 8 is used. Each column reports 'Mean - Median (Standard Deviation)'.

	Classifier	ZM-FRR	EER
KDS-2 Short	*Mahal.*	N/A	N/A
	Man. (scaled)	$0.458 - 0.411(0.438)$	$0.113 - 0.100(0.142)$
	O-Count	$0.272 - 0.050(0.354)$	$0.087 - 0.025(0.111)$
KDS-2 Long	*Mahal.*	N/A	N/A
	Man. (scaled)	$0.238 - 0.0(0.380)$	$0.067 - 0.0(0.109)$
	O-Count	$0.142 - 0.0(0.304)$	$0.049 - 0.0(0.103)$
KDS-3	*Mahal.*	N/A	N/A
	Man. (scaled)	$0.252 - 0.0(0.407)$	$0.06 - 0.0(0.099)$
	O-Count	$0.106 - 0.0(0.252)$	$0.043 - 0.0(0.099)$

False Accept Rate equals the False Reject Rate. The ZM-FRR is calculated by adjusting the threshold to report FRR when the FAR is zero.

We analyzed classification performance for short and long passwords. The results are summarized in Tables 10. The reader can notice that we did not report the results for the Mahalanobis distance measure, which performed well in previous studies. The lack of results is due to the fact that Mahalanobis distance $D_M(\vec{x})$ of a multivariate vector $\vec{x} = (x_1, x_2, x_3, \ldots, x_n)^T$ is computed from a group of m values with mean $\vec{\mu} = (\mu_1, \mu_2, \mu_3, \ldots, \mu_n)^T$ and covariance matrix $\mathbf{S}$, defined as: $D_M(\vec{x}) = \sqrt{(\vec{x} - \vec{\mu})^T \mathbf{S}^{-1} (\vec{x} - \vec{\mu})}$. The covariance matrix $\mathbf{S}$ is of size, $m \times n$, $m \geq n$, where m is the sample size and n is the number of considered variables (features). For $m < n$, the empirical estimate of the covariance matrix becomes singular. In our experiment setup, $\vec{x}$ denotes the keystroke entry being tested while $\vec{\mu}$ denotes the mean of the keystroke entries in the training window. m is the number of keystroke entries in the training window and is restricted to 10, while n is the length of a keystroke entry (i.e. feature vector length). $n = 24$ for short passwords and (at least) 42 for long passwords. As $m < n$, the Mahalanobis distance based classifier could not be used. This natural restriction of the Mahalanobis distance measure is a point of concern when working with longer passwords. Typical password requirements nowadays can be more stringent than the uniform format of long passwords considered here. As password complexity increases, the feature vector length increases. The nature of the Mahalanobis distance requires a large number of training samples to create a user model for keystroke dynamics based classification. For example, training a Mahalanobis distance based classifier using our long password would require at least 42 keystroke entries by each user. In a real-world application, this would imply that keystroke authentication must await for weeks before it is ready for deployment, likely a limiting factor undesirable in most fast pace environments.

The above experiment was replicated on KDS-3. Since all 51 users typed the same password, we randomly paired 25 users with other 25 users to act as

their impostor. As the objective of the experiment is to determine the best distance measure while removing the effects of habituation, we used keystroke entries from the last session (Session 8). This ensures that the data with the least effects of habituation is used. Thus, every distance-based classifier for each of the 50 users was trained using keystroke entries from Window 3 of Session 8, and tested on the Window 4 keystroke entries of that user and his/her randomly paired 'impostor' user. The results for this experiment on this dataset are given in Table 10.

This experiment could not be replicated on KDS-1 since each user in that dataset entered a different password. Thus, the users could not be paired with each other. The results from Table 10 are discussed below.

- Between Manhattan distance and Outlier count, we found that the Outlier Count (z-score) algorithm performs better in all datasets. It was also more stable as evidenced by its comparatively low variance over all users. Our results strongly indicate that even when the effects of habituation are removed, Outlier Count is the better classifier.

- The ZM-FRR of both distance - based classifiers suffered under the short password scenario (in KDS-2) with an average ZM-FRR of 37.1%. Low performance is likely due to the small size of the training set, limited here to 10. However, the purpose of our experiments is to determine variations in typing patterns over time. Therefore, if the size of the training set increases by including early password entries, variance will increase, making the distance measures less reliable.

These results indicate that distance based measures may not be appropriate for keystroke authentication if passwords change often, making the size of available training samples insufficient.

3.7.2 Does Habituation Affect Authentication Performance?

In Section 3.4, we showed that there is a statistically significant reduction in total password typing time due to user learning. We also showed in Section 3.5 that there is a significant reduction in the variance of the total typing time, i.e. the keystroke entries appear to become more uniform. In this section, we would like to ascertain the effect of habituation on the performance of the authentication classifier trained on KDS-2. This is important because the typing time and overall variance are cumulative measures, while authentication is based on the similarities within the multidimensional password keystroke feature vector. Cumulative typing trends may not necessarily result in an improvement in classification performance.

In this experiment, we used two classification algorithms. The first is Outlier Count (z-score), which showed the best performance amongst the unary classifiers (distance measures) in the previous experiment. The second algorithm is the binary classifier based on Random Forest, recommended for keystroke dynamics based user authentication in [6]. Unlike unary classifiers, binary ones

need to be trained using the positive examples (genuine user's password typing) and negative ones (impostor's typing attempts with the same password). KDS-2 is conducive to binary classification because two individuals are given the same pair of passwords. Nevertheless, in operational situations, generation of impostor password keystroke entries would likely have to be automated. Automated generation of impostor keystroke entries for the purpose of training binary classifiers is outside of the scope of this paper. Random forest algorithm is known to be resilient to noise, but it also shows greater performance variation in repeated experiments due to random selection (bootstrapping) of training instances [13]. Random Forest relies on ensemble learning. It generates multiple decision trees that submit their vote to give the classification result.

In KDS-3, all users were assigned the same password. Each user was paired randomly with another user, thus generating 51 random user pairs. The keystroke data from Session 1 was used since the habituation affects the initial session the most. The KDS-1 dataset was not used for the same reasons as mentioned in Section 3.7.1, i.e. the users were assigned different passwords. Due to this, they cannot be paired with another user.

The Outlier Count algorithm was trained and tested in the same manner as described in Section 3.7.1. The classifier was trained using password training instances within Window X of User A and tested on instances from Window Y of users A and A_{pair}. The procedure to train and test the Random Forest (RF) classifier is different, as it requires genuine and impostor data for training. Our RF classifier was trained using Window X of both User A and A_{pair}. It was tested on Window Y of users A and A_{pair}. Random Forest returns the number of votes from internally generated decision trees in favor of classification of an instance in one of the two classes. The ROC curve for the RF classifier is generated by varying the percentage of votes required to make a decision. Voting percentage acts as the threshold in RF classification.

In order to assess the effect of habituation on authentication performance, we set up the experiment as follows. Two classifiers for each password were trained and tested using the window pairs W_{1-4} and W_{4-1}, where W_{A-B} implies that the classifier was trained on Window A and tested on Window B. We calculated the EER for each classifier instance. Thus, we generated 30 classifier instances per password per window pair (30 user pairs) for KDS-2 dataset and 51 classifier instances for KDS-3 dataset.

The EERs of these window pairs were compared against each other to test if there was any statistically significant change in classification performance. We used the Wilcoxon Signed Rank test for this analysis because we could not assume normality in this distribution. If there is no difference between the keystroke entries in the training window and the testing window, then the classifier should not show significant difference in performance when the training and test sets are reversed. If habituation in typing patterns exist, there is likely to be a difference in performance of authentication classifiers trained and tested on reverse windows.

The results of the experiment are shown in Table 11. We reiterate that notation $W_{A-B} : W_{C-D}$ means that the EER of the classifier trained on Window

Table 11: Statistical significance of the difference in classifier performance when trained and tested using different data. The comparison is made between classifiers built and tested on $W_{1-4} : W_{4-1}$. The classifiers tested are Outlier Count (OC) and Random Forest (RF).

	KDS-2 (Short)		**KDS-2 (Long)**		**KDS-3**	
	OC	RF	OC	RF	OC	RF
p	0.0427	0.4562	0.0001	0.0025	0.0023	0.2451
Sig.	YES	NO	YES	YES	YES	NO

A and tested on Window B is compared to the EER of the classifier trained on Window C and tested on Window D. The results lead us to the following considerations.

- For the long passwords (KDS-2), there is a statistically significant difference between classification outcomes trained and tested on W_{1-4} versus the classification outcomes of the same two algorithms trained and tested using the reverse windows W_{4-1} . This shows that the Window W_1 contains keystroke entries which differ from those in Window W_4 in terms of the authentication classification performance they generate.

- The EER is higher (i.e. the classifier performs worse) for the classifier W_{4-1}, which is trained on Window W_4. This is because the window W_4 contains password keystroke entries which exhibit lower variance compared to W_1. Therefore, the classifier is more likely to misclassify entries from W_1 after being trained on W_4. The strength of rejection of the null hypothesis is further corroborated by the magnitude of rejection in total typing time and the variance of the total typing time shown earlier in Sections 3.4 and 3.5.

- For the short passwords in KDS-2, the Outlier Count shows a significant difference in EER. The same cannot be said of Random Forest. We believe the reason for the lack of significance in classification performance of random forest is due to the reduced level of habituation in short passwords. The magnitude of reduction in total typing time between W_1 and W_4 was around 0.1 seconds for short passwords and between 1.5 and 3.0 seconds for long passwords. Further, random forest classifier is not as sensitive to "noise" in the training data compared to a distance measure.

- The above trend was observed in KDS-3 dataset too where Outlier Count showed a significant difference in EER, but Random Forest did not. The magnitude of reduction in total typing time between W_1 and W_4 in this dataset was around 0.4 seconds. This seems to be the major contributor to the lack of significant difference in performance for the Random Forest classifier.

3.7.3 How to Accommodate Habituation in Keystroke - Enhanced Authentication?

In Section 3.7.2, we observed that training the classifier with keystroke entries of a user habituated to his or her new password and testing the classifier with the entries made prior to habituation shows a drop in performance when compared to the situation in which the classifier is trained with early password entries. Obviously, the W_{4-1} train - test scenario is illogical for practical implementations in which keystroke entries are made available in the chronological order. It was used only to prove the point related to the impact of habituation on authentication performance of keystroke dynamics systems.

In this section we analyze and compare the performance of practical authentication classification alternatives. In this case, practical means that the keystroke entries used for model training precede the ones on which the model testing. We also limit our attention to the performance of random forest (RF) classifier, which consistently outperformed the best distance measure in our experiments. We constructed multiple instances of the RF classifier for every user / password combination. We used window pairs W_{1-4}, W_{2-4} and W_{3-4} for training and testing, where W_{A-B} denotes that the entries in Window A are used for model training while Window B are used for classification testing. The procedure to train the RF classifier was explained in Section 3.7.1. In this experiment, we used the default simple majority voting scheme to determine an authentication outcome (the ensemble of decision trees within RF vote to decide the class of a feature vector). We measure the classifier's performance based on its accuracy. The classifier accuracy refers to the number of keystroke entries in the test set that are correctly classified across all users and passwords.

The design of KDS-2 facilitates a pairwise comparison wherein two users are assigned the same credential. However, due to the nature of KDS-3 wherein all users are assigned the same credential a pairwise comparison cannot be made unless pairs of users are randomly matched to each other. The KDS-3 dataset contains 51 users, a total of 25 user pairs out of a possible 1275 user pairs was selected for this study.

In order to compute if there is a change in classifier performance emanating from the selection of the window used for training, we compared the the accuracy of different classifier instances. To reiterate, accuracy refers to the percentage of correctly classified instances. For example, we compared the performance of $W_{1-4} : W_{2-4}$, i.e., measured the difference in accuracy when classifiers are trained using W_1 as compared to W_2 while using an identical test set W_4.

The following hypothesis is tested using Friedman's Test:

NULL: - The authentication accuracy does not change with different training windows.

ALT: - The authentication accuracy changes (improves) when the keystroke training entries and test entries are chronologically close.

The results of this test are shown in Table 12 for KDS-2 and KDS-3. We draw the following observations:

Table 12: Statistical comparison of habituation on classifier performance (RF) for passwords in KDS-2 and KDS-3 at $\alpha = 0.05$

Friedman's test	*Post-hoc analysis*			
		$W_{1-4} : W_{2-4}$	$W_{2-4} : W_{3-4}$	$W_{1-4} : W_{3-4}$
KDS-2 Short *p=0.18*	*p*	-	-	-
	Sig.	NO	NO	NO
	Avg. Accuracy	W_{1-4}**: 88%**	W_{2-4}**: 90%**	W_{3-4}**: 90%**
KDS-2 Long *p=0.011*	*p*	0.47	0.020	0.0416
	Sig.	NO	YES	YES
	Avg. Accuracy	W_{1-4}**: 83%**	W_{2-4}**: 83%**	W_{3-4}**: 91%**
KDS-3 *p=0.003*	*p*	0.0046	0.2382	0.0021
	Sig.	YES	NO	YES
	Avg. Accuracy	W_{1-4}**: 86%**	W_{2-4}**: 92%**	W_{3-4}**: 93%**

- For *short passwords* in KDS-2, the reduction in variability in the typing patterns of keystroke entries between the windows of observation did not lead to a statistically significant improvement in the accuracy, which measures authentication performance. Using the later keystroke entries for model training is no more effective than using early keystroke entries. This result corroborates earlier experiment in which Random Forest algorithm showed no significant difference between classifiers' EER based on W_{1-4} and W_{4-1}. For short passwords, the null hypothesis cannot be rejected.

- For *long passwords* in KDS-2, the classifier accuracy improves significantly when later keystroke entries are used to train / update classification model. Later entries exhibit lesser typing rhythm variation and are more uniform. This translates to better accuracy. Again, this result corroborates the experimental outcomes from Section 3.5. In most cases, the null hypothesis is rejected.

- For passwords in KDS-3, the classifier accuracy improves significantly when later keystroke entries are used to train / update classification model in Sessions 1, 6, 7 and 8. In most cases, the null hypothesis is not rejected. This phenomenon is due to the nature of the dataset where a user is asked to submit a total of 50 entries per user session. This effect is further highlighted in Fig. 12. We notice that over time (increasing Session numbers) the overall accuracy shows a gradual increase. Moreover, the difference in performance between training and testing with with temporally close and distant values decreases over time.

Therefore, authentication classifiers trained on the keystroke entries that reflect habituation perform significantly better than the ones trained on early keystroke long password entries. This does not mean that a classifier trained on

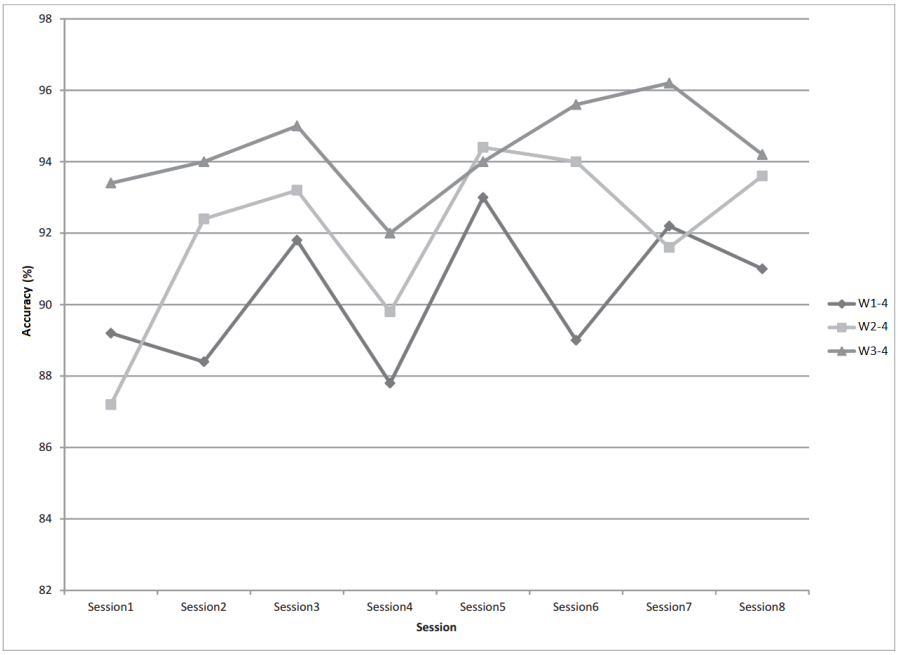


Figure 12: Effect of Habituation on classifier accuracy in KDS-3. In the legend,$WX - Y$ indicates a classifier trained on Window X and tested on Window Y.

all prior keystroke entries will necessarily perform worse. We analyze this question in the next experiment. Current experiment compared models which used the same size of the training and test entries, thus eliminating the confounding factor of performance evaluation attributed to the variations in the size of the training and test datasets. As practitioners develop keystroke-dynamics based authentication systems, they should remain cognizant of the temporal distance between training and test data. In the next section, we further explore the performance of additional practical authentication models for keystroke dynamics.

3.8 RQ7: Which training model best leverages the effect of habituation to develop an effective keystroke dynamics authentication system?

Section 3.7.3 showed that keystroke entries temporally close to authentication instances (test instances) should be used training. However, authentication models should find a balance between effectiveness and the ease of deployment. In this section, we analyze techniques to create most effective authentication

Table 13: Performance metrics of the four authentication models: accuracy (%) are followed by their variance (%) over all users, in parentheses

Password Type	Model 1	Model 2	Model 3	Model 4
Short	90 (2.00)	93 (0.22)	96 (0.15)	91 (2.27)
Long	87 (3.27)	94 (0.23)	97 (0.11)	93 (1.70)

classifiers. Each model describes a certain approach to train the classifier. We test each model using KDS-2 dataset and report the performance results in terms of the accuracy and the variance of accuracy across all users. Each authentication model is developed using the Weka implementation of the Random Forest algorithm with an ensemble of 500 decision trees and the majority voting scheme.

The results of the four models are shown in Table 13 and discussed below.

3.8.1 Model 1 - Train on the First 10 Entries

In this static model, the classifier instance for a user is trained using her / his first 10 password entries. The remaining 30 entries are used for testing. This is the simplest model to develop, as it is never updated. Overall, the model achieves a 90% accuracy for short passwords and 87% accuracy for long passwords. Given the limited training data available, this may be acceptable. In Sections 3.4 and 3.5**,** the effects of habituation have been shown to have a stronger effect on classifier performance for long passwords. Hence, a reduced performance for long passwords is expected due to the use of early keystroke entries for training.

3.8.2 Model 2 - Train on the 10 Most Recent Entries

In this a dynamic, continually updated model. The 10 most recent entries are used to train the classifier. In other words, every test entry uses the classifier trained on the most recent 10 entries. The classifier is updated every time the user submits a new entry. This model performs better than Model 1. The accuracy improves by 3% for short passwords and 7% for long passwords. More importantly, the variance drops by a factor of 10. Thus, a dynamic model that retrains itself using the 10 most recent entries is more effective than the static model. The drawback of this model is that it needs constant updating, which incurs computational overhead every time someone attempts to log into the system protected by keystroke dynamics authentication.

3.8.3 Model 3 - Train with All Prior Entries

In this model, all prior entries submitted by the user are used to train the authentication model. All the remaining entries in KDS-2 are used for testing. In order to keep comparisons with other models fair, this model is trained starting

with the 11^{th} password entry such that the size of the testing set is the same as in Models 1 and 2. The results show that this model further improves Model 2. The accuracy for short and long passwords show a 3% improvement. Furthermore, the variance decreases by additional 50% compared to Model 2. A dynamic model that trains on all prior user entries seems to provide the most effective authentication. However, the results of this model are confounded by the different size of the training set compared to Models 1 and 2, which use 10 training instances only. It can be argued that the improvement in performance is based on the volume of data used to train the classifier. This model has the highest computational overhead due to the constantly increasing size of the training set. Furthermore, it is important to remember that our results from Section 3.7 indicate that changes in the users' typing patterns significantly affect authentication performance. Therefore, over the long run, it may be counterproductive to use early entries to train the classifier. Unfortunately, our datasets are not large enough to analyze this issue.

3.8.4 Model 4 - Train From the Most Recent Window of 10 Entries

This model is similar to Model 2. However, instead of updating the model every time the user submits a keystroke entry, this model is updated once every 10 keystroke entries. Thus, the first iteration trains using entries $1 - 10$ and is tested on the entries $11 - 20$. The classifier is then rebuilt using entries $11 - 20$ and tested on entries $21 - 30$, and so forth. This model alleviates some of the computational overhead caused by the rapid retraining of authentication models after every new password entry. This model performs better than Model 1. However, it does not match the performance of Models 2 and 3. Although it is very similar to Model 2, the variance of Model 1 increases by a factor of 10. We attribute increased variance to the delayed model update in the phases of active habituation.

3.8.5 Conclusions

Our results indicate that Model 3, which uses all prior entries for training offers the best authentication performance. However, this model is not the best to use in a real-world system because of the large volume of training data required. Therefore, training the classifier using only the most recent user-specific keystroke entries (Model 2) is practical and beneficial. We have shown in Section 3.4 and 3.5, the user's typing pattern changes significantly over time. It appears fair to observe that the best models should be retrained using only the most recent keystroke entries while discarding the older ones. Based on the authentication performance and ease of classifier generation compared to other models, Model 2 offers the best balance between authentication performance and computational overhead.

3.9 Leveraging event sequences to create better keystroke dynamics based authentication systems

3.9.1 Motivation

User model development in keystroke dynamics is based on two primary attributes: hold time and delay time. As described in Section 2.1.1, hold time refers to the duration of a keypress and delay time refers to the latency between two consecutive keypresses. Due to their inherent nature, hold and delay times are dependent on the user's typing speed. A fast typist exhibits shorter hold and delay times compared to a slow typist. This leads to better discriminability between users with different typing speeds. However, their disadvantage is that users with similar typing speeds are easily confused by the classifier. In order to overcome this problem, we propose a new type of attribute for keystroke dynamics called event sequences. This section is organized as follows: Subsection 3.9.2 lays out the concepts, terms and representations used. Subsection 3.9.3 provides a brief overview of the various keyboard layouts currently in use. Subsection 3.9.4 elaborates on event sequences and its causes. Subsection 3.1.2 provides a description of the data set collected for the purposes of this study. The effectiveness of user authentication performed using event sequences is experimentally analyzed in Subsection 3.9.5.

3.9.2 Terms and representations

For the purposes of this section, the character keys (defined in 3.9.3) are denoted using the character that they output on the screen when using the ISO-US English layout. All keys other than the character keys are denoted using the abbreviations shown in Table 14.

We define a *string* as the string of characters that are to be created using the keyboard. A string can be a single character, a word, a sentence or a paragraphs or a set of these.

A *keystroke* is defined as the act of pressing and releasing a key. It consists of two events: Key down (*KeyDn*) and Key up (*KeyUp*). The KeyDn event of any key X is denoted as X_d and the KeyUp event as X_u. If the KeyDn and KeyUp events of a key occur consecutively, it is denoted as X_{du}.

Keystroke dynamics data is derived from the time stamps of the KeyDn and KeyUp events. The primary attributes that are used to derive other metrics are hold and delay times. The delay time is the time between two successive key presses, i.e. between two successive KeyDn events. The hold time is the time for which a key is pressed, i.e. the time elapsed between the KeyDn and KeyUp events for a key.

We also define a notation to represent the event schema and the event sequence. The *event sequence* is the temporal sequence of all KeyDn and KeyUp events performed to type a string. This includes the KeyDn and KeyUp events

of character keys and of special keys (Caps Lock, Left Shift, Right Shift, etc) that are used to modify the character key output. The *event schema* is a generalized representation of similar event sequences. The following example is an illustration:

Suppose a user types the string $W\underline{E}hat$ where the underlined E indicates that the user erased it using the backspace (Bk) key. One possible event sequence for this string is $\{Rs_d\, W_{du}\, Rs_u\, Ls_d\, E_{du}\, Ls_u\, Bk_{du}\, h_{du}\, a_{du}\, t_{du}\}$. Note that the Right Shift key (Rs) is used to type W while the Left Shift key (Ls) is used to type E. Different Shift keys could be used to type the uppercase letters, resulting in multiple event sequences for this string. All of these event sequences can be represented using the event schema $\{Sh_d\, W_{du}\, Sh_u\, Sh_d\, E_{du}\, Sh_u\, Bk_{du}\, h_{du}\, a_{du}\, t_{du}\}$, where Sh denotes any Shift key.

3.9.3 Keyboard Layouts

A computer keyboard consists of character keys, modifier keys for altering the functions of character keys, navigation keys for moving the text cursor on the screen, etc. Extended keyboards also have a numeric keypad to facilitate calculations. A keyboard layout refers to the arrangement of the keys, key labels or key mappings on a keyboard. Keyboard layouts can be categorized based on three criteria:

1. Mechanical layout refers to the physical keys and their placement on a keyboard. Most keyboards use one of three mechanical layouts. The layouts include:

   (a) ISO layout is dictated by the ISO/IEC 9995 framework [39].

   (b) ANSI layout follows the ANSI-INCITS 154 – 1988 framework [3].

   (c) JIS layout follows the Japanese Industrial Standard JIS X 6002 : 1980 framework [40].

2. Visual layout refers to the arrangement of the legends (labels, markings, engravings) that appear on the keys of a keyboard.

3. Functional layout describes the software-based association of a key to character(s) for all keys on the keyboard. The functional layout determines the key sequence required to enter a character. For example, one functional layout outputs a character X by pressing a certain key, while another layout would do so using another key (possibly with a modifier key such as a Shift). Advanced users can also create custom functional layouts using off-the-shelf software [70]. Thus the functional layout determines the keys to be pressed to output a character.

A functional layout organizes characters that the keyboard can generate into levels. There are generally two levels on a keyboard. However, the ISO standard also allows for the level 3. Characters at Level 1 are accessed by simply pressing the corresponding key. Characters at higher levels are accessed using modifier

Table 14: Key abbreviations used in Section 3.9

Key	Notation	Key	Notation
Alt	Al	Right Shift	Rs
Left Alt	Lt	Left Shift	Ls
Right Alt	Rt	Either Shift	Sh
Caps-lock	Cl	Numpad $0-9$	N0-N9
		Numpad symbols	N+, N-, N/, N*, N., N=

Table 15: Type I and II characters on an ISO-US layout

Punctuation		Symbols				
Type I	Type II	Type I	Type II			
;	:	[	@	^	_	<
’	”	]	#	&	{	>
,	!	\	$	(	}	
	?		%	)	|	

keys. Level 2 characters are accessed using the Shift key and Level 3 characters (in ISO layout) are accessed via the AltGr key (The ISO layout renames the right Alt key to AltGr) [53]. In addition, the Caps-lock key can be used to change the case for some characters that are dual-case, e.g, the Latin alphabets A-Z on an ANSI-US layout. The key types relevant to this study are enumerated below. These keys are present on all mechanical layouts:

1. Character keys: These are the keys denoted by letters A-Z, digits $0-9$, punctuation marks, and symbols on a US layout.

2. Modifier keys: These are the Alt, and Shift keys. A set of these is located on either side of the keyboard (with the right Alt key replaced by the AltGr key in an ISO layout). We also include the Caps-lock key in this category.

3. Numeric keypad: The keypad comprises of number keys $0-9$ and the math operator keys (+, -, /, {*}, .).

3.9.4 Causes of Variations in Event Sequences

The number of event sequences for a given string is determined by the keyboard's functional layout, the variations in use of modifier keys, the types of characters in the string and their order. For clarity, we use the ANSI-based US functional layout to explain the effect of each of these variables on the number of possible event sequences.

The effect of change in modifier key usage

The event sequence of prior character(s) affects possible event sequences to type the subsequent character in a string. For example, consider a simple 2 letter string *'KA'*. Assuming that the Caps-lock is initially disabled, the user can type the string using 8 different event sequences:

1. $\{Sh_d\, K_{du}\, A_{du}\, Sh_u\}$ - This event schema represents two distinct event sequences where the user can use the Right Shift (Rs) or the Left Shift (Ls) key.
2. $\{Sh_d\, K_{du}\, Sh_u\, Sh_d\, A_{du}\, Sh_u\}$ - Each of the two characters can be typed using either of the two Shift keys resulting in 4 event sequences.
3. $\{Sh_d\, K_{du}\, Sh_u\, Cl_{du}\, A_{du}\}$ - Using either of the two Shift keys, the string can be typed in this schema using 2 distinct event sequences.

However, if the first character is typed after first enabling the Caps-lock, it can be entered as $\{Cl_{du}\, K_{du}\, A_{du}\}$. Thus, the string KA can be typed using 9 different event sequences.

The effect of string content

Another reason for the variation in the number of event sequences for a fixed string is the contents of the string itself. In order to understand this, the characters need to be divided according to the number of event sequences that they can possibly be typed through. For the ANSI-US layout, various characters can be typed using the character and modifier keys. They can be classified into 3 types based the number of event sequences that they can emerge from:

1. Type I characters can be typed using only one event sequence irrespective of the characters before or after. Table 15 lists these for the US layout. For example, the semicolon (;) can only be typed with the event sequence $\{;_{du}\}$ even when it occurs with other characters, e.g. $\{Sh_d\, A_{du}\, Sh_u\, ;_{du}\, w_{du}\}$.
2. Type II characters can be typed with just one event sequence only when they are typed in isolation, e.g. the character '&' as in $\{Sh_d\, \&_{du}\, Sh_u\}$. However, when typed together with other characters, they can be typed using different event sequences, e.g. $\{Sh_d\, A_{du}\, \&_{du} Sh_u\}$ or $\{Sh_d\, A_{du}\, Sh_u\, Sh_d\, \&_{du} Sh_u\}$. Table 15 lists the Type II characters for the US layout.
3. Type III characters include all those not in Type I and II. These can be entered using more than one event sequence both when isolated or when part of a string. On the ANSI-US layout, alphabets A-Z (lowercase or uppercase) may be entered either by first enabling Caps-lock such as $\{Cl_{du}\, A_{du}\}$ or by using a Shift key, e.g. $\{Sh_d\, A_{du}\, Sh_u\}$. Numbers $0-9$ can be typed using the standard number keys such as $\{6_{du}\}$ or using the numeric keypad e.g., $\{N6_{du}\}$. Mathematical operators $(+ - */)$ can be typed using either the numeric keypad such as $\{N+_{du}\}$ or using the number keys e.g., $\{Sh_d\ +_{du}\ Sh_u\}$.

Table 16: The number of possible event sequences changes due to the string length or the order of characters within the string.

String Length	**String 1**	**String 2**	**Event sequences**	
			String 1	*String* 2
1	K	4	3	2
2	KA	4!	11	4
3	KA:	4!K	24	20
4	KA:a	4!Ka	30	40
4	KAa:	4!aK	38	36
5	KA:a3	4!Ka5	60	80
5	KAa:3	4!aK5	76	72

The effect of character order

Table 16 demonstrates how the number of possible event sequences varies based on the character types in the string and their order. Notice that strings with the same length or same characters can have different number of possible event sequences due to the change in character order.

3.9.5 Experimental Analysis of Event Sequences

Having demonstrated the variations in event sequences for any given string, we proceed to demonstrate observations from a realistic usage scenario. Further, we want to learn whether different event sequences have negative impact on keystroke - based user authentication. For the experimental analysis, we used KDS-2 dataset described in Section 3.1.2. In Sections 3.9.6-3.9.8, a user-password combination in the dataset is referred to as a *subject*. A *subject pair* refers to the user pair when they are typing the same password. There are 60 subject pairs in the study (each of the 15 user pairs is given two passwords, each can be used as a genuine / impostor attempt).

We set up the experiments to answer three research questions:

1. Do pairs of users employ similar event sequences when typing the same string over a period of time?

2. Is there any correlation between typing proficiency and event sequences exposed by a user?

3. What is the effect of time and habituation on the event sequences of a user?

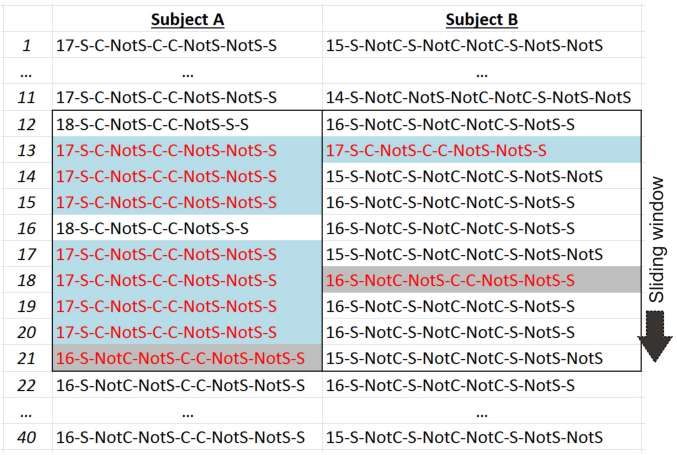


Figure 13: Computing the similarity between ant two paired subjects with respect to event signatures: For the current sliding window, user A has 8 entries with the same event signatures as user B, while user B has 2 entries that are similar to user A. The non-match score for user A is 0.2 while it is 0.8 for user B.

3.9.6 RQ8: Do pairs of users employ similar event sequences when typing the same string over a period of time?

In this experiment, we quantitatively measure the similarity amongst paired users with respect to the event sequences observed while they typed their passwords. To do so, all keystroke entries in the dataset were first represented with an event signature as shown in Fig. 13. For each subject, there are 40 event signatures ordered temporally. Each signature represents one password entry. Event signature is an abbreviated version of the event sequence described in Section 3.9. It does not convey the same amount of information as the event sequence. However, it is easier to construct from the data and, given the knowledge of the specific string, it can adequately represent every unique event sequence for that string.

The event signature consists of concatenated tokens, as shown in Fig. 13. The first token is numeric and represents the number of events in the event sequence for that keystroke entry. The remaining tokens indicate the user's decision to use or omit using the Shift or Capslock key at instances where it was necessary to do so. These instances occurred primarily when the case of the next character to be typed was different. The remaining tokens are one of four types: *S*, *NotS*, *C*, *NotC*. '*S*' and '*C*' indicate that the user used the Shift or Capslock key, respectively, while '*NotS*' and '*NotC*' indicate that he/she did not. As mentioned in Section 3.1.2, the dataset does not contain information on whether the right or left Shift key was used.

A sliding window is used on each subject pair (User$A$-Password$X$ and UserA_{pair}-Password$X$) to calculate the percentage of event sequences that are unique to each subject. This technique is illustrated in Fig. 13. The idea behind this experiment is to simulate a scenario where an impostor has access to the genuine user's password and attempts to use it. In this sense, each user in a pair acts as the impostor to the other while they are compared for similarity on the basis of their event signatures only.

The sliding window technique generates 31 non-match scores for every subject. Each non-match score indicates the percentage of entries in the window that are distinct from the paired subject with respect to their event signature, i.e. a higher non-match score indicates less similarity between subjects. The mean and median non-match score for each of the 60 subjects was computed and the distribution of these scores is tabulated in Table 17. The results can be summarized as follows:

1. Across all subjects, event signatures were distinct amongst subjects in ˜61% of the entries in any given window.

2. Table 17 also shows the percentage of subjects that had a mean and median non-match score of 100%, $>$ 75% and $>$ 50% for any given window. For ˜31% of the subjects, event sequences are completely distinct from those of the paired subject in every single window, i.e. these subject pair did not have a single similar event sequence throughout the study.

3. This percentage is much higher (42.8%) when considering the median non-match score. This is because although subjects sometimes displayed high but less than perfect scores for some windows, calculating the median score for a subject over all windows would still give a median score of 10. Thus, a larger percentage of users appear to have a perfect score.

4. Another important observation is the degree of variation in similarity between subjects. 62% of the subjects showed a score greater than 50%. For this group, the mean score was 91.3%. The mean recall rate for subjects with a score less than 50% was 11.5%. This uneven distribution of scores is further illustrated in the histogram in Fig. 14.

The experimental results offer empirical evidence that users display considerably different patterns when typing. Only 30% of the subjects had non-match scores between 10% and 90%. The distribution of non-match scores indicates that the variations in keystroke sequences create a distinguishing factor in the majority of cases. Furthermore, as we previously mentioned, this dataset does not contain information on which Shift key (right or left) was used. Based on results reported by other authors on the importance of Shift key behavior [6], we believe that using this information would distinguish users even further and boost the non-match scores.

Table 17: Distribution chart for recall rates using event sequence signatures

	% of users	
Mean recall rate	Median	Mean
$= 100\%$	42.3%	30.8%
$> 75\%$	48.1%	46.2%
$> 50\%$	61.5%	61.5%
Overall	60.5%	60.8%
Recall when $> 50\%$	91.3%	89.5%
Recall when $< 50\%$	11.5%	14.9%

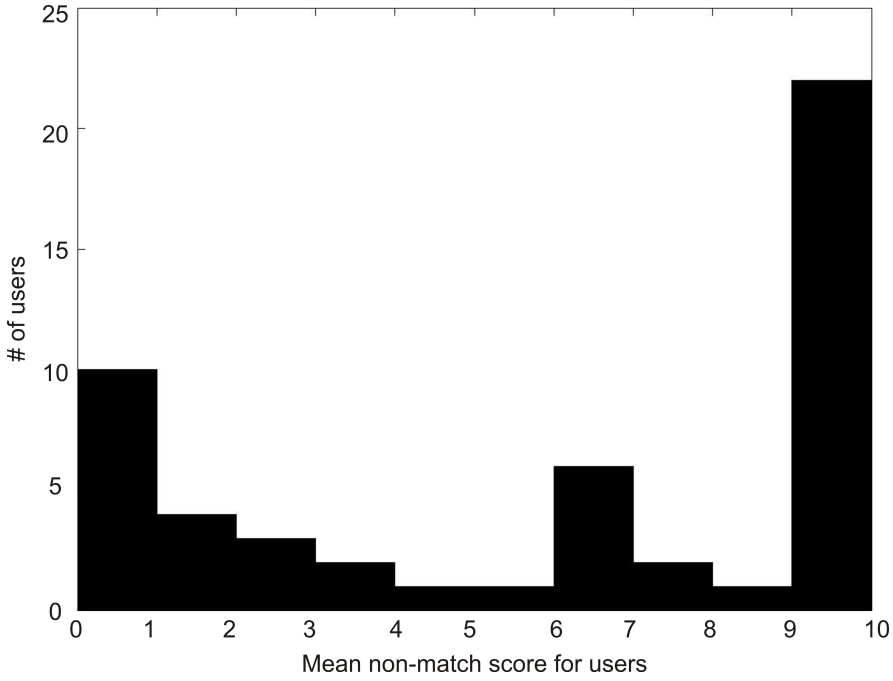


Figure 14: This distribution illustrates the number of users exhibiting specific non-match scores, i.e. the degree of similarity to their paired users. A higher score indicates less similarity.

Table 18: Calculating correlation between typing proficiency (TT stands for Typing Time) and event sequence usage

Subject #	**Grp A users**		**Grp B users**		**Difference**	
	TT	*Recall*	*TT*	*Recall*	*TT*	*Recall*
1	9348	0.975	7158	0.975	2190	0
2	4299	0.075	5979	0.75	−1680	0.025
...	...	...	...	...	...	...
60	8657	0.75	4693	0.925	3963	0.175
Correlation					**0.016**	

3.9.7 RQ9: What is the Correlation Between Typing Proficiency and Event Sequence?

Upon observing the uneven distribution of recall rates for the subject pairs, we hypothesized that there may be a relation between the user's typing proficiency and event sequences used. We theorized that fast typists tend to type in a manner that may be dissimilar to slower typists and that this may be the cause of the uneven distribution of non-match scores observed in the previous experiment. Therefore, the objective of this experiment is to determine if there is any correlation between the typing proficiency of users and the event sequences they might use. In order to test this, we first computed the difference in median typing time and, additionally, the difference in non-match score for each subject pair, as shown in Table 18. We computed the correlation coefficient between these two sets of values. This value quantifies the correlation between the difference in typing proficiency in a subject pair and the dissimilarity between their event sequences.

The analysis indicated a Pearson's correlation coefficient of 0.016. This low correlation value means that for a given pair of users, there is little or no correlation between the difference in typing proficiency and a preference for certain event sequences over the others. In other words, users do not seem to prefer certain event sequences over others based on their typing proficiency. Event sequences are independent of user's typing-proficiency. This result suggests that event sequences provide a discriminatory authentication dimension that is not dependent on user typing proficiency, but rather on other behavioral characteristics of each user. Upon examining the dataset further, we observed that some hunt-and-peck users would use the same event sequence as their paired user who were proficient typists. While we could not identify the reasons behind this observation, the users do not seem to prefer certain event sequences over others due to their typing proficiency. Therefore, allowing users to type a string in the manner they wish to should be helpful in differentiating between users even if they have similar typing proficiency. However, almost the entire keystroke authentication literature ignores typing sequences.

3.9.8 RQ10: What is the Effect of Time and Habituation on the Event Sequences Used by a User?

In [73], we showed that as users type a string more frequently, they exhibit habituation that reduces the time required to type the string. We theorized that, in a similar manner, users will tend to use fewer event sequences as they habituate to typing a given string.

In order to analyze this behavior, we used the event signature representation of keystrokes described in Section 3.9.6. Using a sliding window of ten entries, the similarity in event signatures was measured using a similarity score S that assigns a higher score when fewer variations within event signatures are observed. The similarity score is described in more detail below. Since every subject contributes 40 entries, using a sliding window of 10 entries gives 31 similarity scores as shown in Steps 1 and 2 in Fig. 15. As shown in Step 2 in Fig. 15, scores denoted 2 to 31, for each subject, are split into 3 windows of equal length and the median similarity score for each window is calculated. Thus, each of the 60 subject pairs contributes a median similarity score for Windows 1, 2, and 3.

Our objective is to measure if there is a significant difference in the similarity scores of any of the three windows. This would indicate that users' typing pattern stabilizes and exhibits fewer variations in event sequences over time. To test this hypothesis, we used the Friedman's test.

Friedman's test

The Friedman test is a non-parametric test for treatment differences in a randomized complete block design [25]. It is used to test for differences between three or more paired groups when the dependent variable being measured is ordinal. The groups in this experiment are the windows. The N subjects and k windows are considered separate independent variables in the analysis. The test statistic for the Friedman's test is a Chi-square with $(k-1)$ degrees of freedom. Thus, the hypotheses for comparison across the windows are:

H0: The distributions are the same across the windows

H1: The distributions across the windows are different

Here, 'distribution' refers to the distribution of similarity scores in any given window. As shown in Fig. 15, for a set of N subjects, 3 groups (windows) are being compared to test if there are differences in similarity scores between any two. The values (similarity scores) across each row are rank-ordered. The resulting ranks are then summed for each column. The null hypothesis asserts that there is no difference amongst the 3 groups. Therefore, the sum of ranked scores in each column R_j should approximately be $\frac{k(N+1)}{2}$. To measure the degree to which each observed sum of ranked column varies from the null hypothesis value,

the Chi-square statistic χ_r^2 for Friedman test is calculated as

$$\chi_r^2 = \frac{12}{Nk(k+1)} \sum_{j=1}^{k} R_j^2 - 3N(k+1)$$

The results of the Friedman test are defined by the parameters χ_r^2 and p. The test statistic χ_r^2 is distributed according to the χ_r^2 distribution with $k-1$ degrees of freedom. The value p defines the probability that the calculated χ_r^2 value will be obtained if the null hypothesis were true. In our experiment, we chose a significance level of $p = 0.05$, i.e. we reject the null hypothesis if $p \leq 0.05$.

The Friedman's test only provides a test to determine if there is a difference in the distribution of similarity scores amongst the windows. It does not provide any information on which pair(s) of windows are different. For this information, a post-hoc analysis is carried out using Wilcoxon signed-rank tests for multiple comparisons between the treatments. A Bonferroni correction is applied to adjust for the multiple comparisons that are being made [11]. We refer the reader to [73] for a detailed explanation of the Wilcoxon test in this scenario.

Similarity score S

The similarity in user typing pattern was measured over a window of consecutive entries using a score. This similarity measure S is formulated as,

$$S = \frac{N_{common}}{N_{seq}}$$

where N_{seq} is the number of different event sequences in the window and N_{common} is the number of times the most common event sequence was typed. For example, consider the window for Subject 1 shown in Fig. 15. Here $N_{seq} = 2$ and $N_{common} = 6$ and thus $S = 3$.

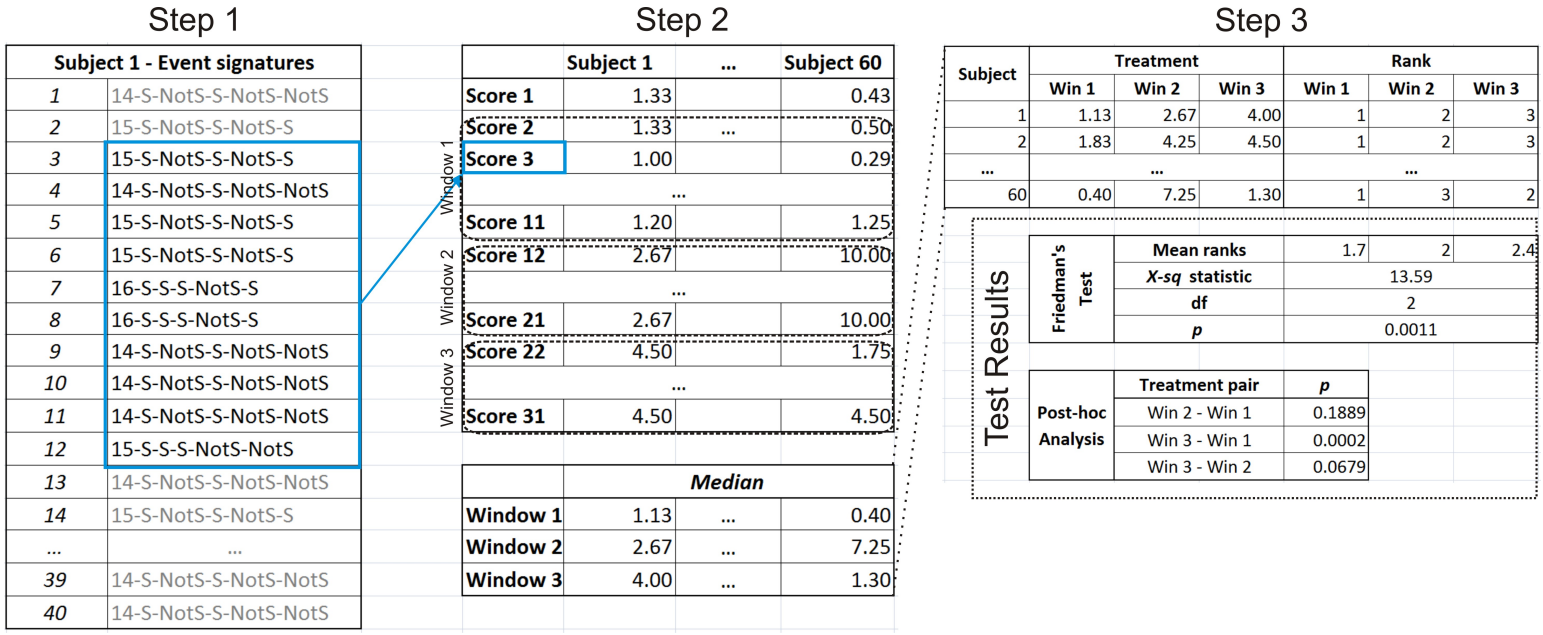


Figure 15: Procedure for testing user habituation using Friedman's test

Results

The procedure and calculated probability values for the Friedman's test are given in Step 3 in Fig. 15. The p value for the χ_r^2 statistic was calculated to be 0.0011. This indicates a significant probability that the event sequence pattern of users changed over time. Additionally, the post-hoc analysis indicated $p = 0.0002$ for Window3-Window1 and $p = 0.0679$ for Window3-Window2. This means that there is a significant difference between event sequence patterns of Windows 3 and 1, and a near significant difference (the value of p slightly higher that 0.05) between the subsequent Windows 3 and 2.

The results reveal that entries that are typed later are far more representative of the user as compared to earlier entries. The implications of these results inform authentication model development and update strategies for keystroke dynamics systems. These results reinforce those presented in [73] where user habituation in the context of hold times and delay times was first analyzed. When combined, these results show that in the course of an acclimation period, the user's typing profile (based upon hold time, delay time, total time, event sequence used, etc.) changes significantly. Therefore, keystroke dynamics based authentication that relies on complex strings would see improved performance if users are asked to train on a string before accepting it as a password. Such training would stabilize the event sequence too. Furthermore, to leverage the discriminatory power of keystroke dynamics and event sequences it is necessary to discard early password entries whenever more mature ones become available.

3.9.9 Event sequence usage in publicly available datasets

The concept of event sequences that is introduced in this work is a novel research idea although it is fairly intuitive. Our survey for research on it did not yield any related work. We refer the reader to Banerjee et al. [5] for a comprehensive survey of the keystroke dynamics research. The survey shows that researchers have made significant efforts in developing better performing algorithms. However, the data used for research has not stepped out of the early experimental bounds. Table 19 summarizes the type of strings for which the keystroke dynamics data was collected and made publicly available. As can be seen, current research has been limited in attempting to leverage the differences caused by strings containing both uppercase and lowercase letters. In the case of Montalvao et al. [56], one of the databases contains free text data which does contain strings with uppercase and lowercase letters. However, no information on the modifier keys exists in any of the datasets. While Killourhy and Maxion [46] used a complex string, an analysis of the dataset shows that the users were restricted to typing the strings using only one event sequence.

Bartlow and Cukic [6] showed that using data from the Shift key modifier in passwords resulted in an improved classifier performance. The passwords used were of the same form as those used in this paper. However, as has been explained in this work, such a string can be typed using many different event sequences. The authors did not discuss if and how the users and impostors were

Table 19: Characteristics of current publicly available keystroke datasets

Dataset	Words used	Remarks
Montalvao et al. [56]	"*chocolate, zebra, banana, taxi.*", "*computador calcula.*"	Lower case string. Single event sequence.
	Free text	Contains uppercase letters but no shift or Caps-lock data
Killourhy and Maxion [46]	*".tie5Roanl"*	Single event sequence
Giot et al. [28]	"*greyc laboratory*"	Single event sequence
Allen [2]	"*pr7q1z*", "*jeffrey allen*", "*drizzle*"	Lower case string, single event sequence.
Bello et al. [7]	Paragraphs from Spanish translation of: *One Thousand and One Arabian Nights* and *War and Peace*	Lower case paragraph, case insensitive. Quotes, dashes, hyphens removed. Users were allowed to make mistakes but researchers used digraphs for analysis

restricted, if at all, towards uniform event sequences.

3.10 Threats to Validity

3.10.1 Threats to Internal validity

1. **Maturation** - In all data collections, users may have experienced boredom while typing the same passwords repeatedly. This problem was addressed by asking the users to submit as many entries as they were comfortable within each session. The duration of the study also allowed sufficient time for the subjects to submit the required number of entries. The users habituated to the password over time. This effect has been a key part of our study and, hence, not a threat to its validity.

2. **Instrumentation** - The keyboard type was not controlled during the collection in KDS-1 and KDS-2. Different keyboards may exhibit differing hold and delay times based on their layout and comfort level. Additionally, users may have used different keyboards while entering the same password leading to additional variation in their typing patterns. This threat to validity was reduced by monitoring the IP address and asking the user to specify their keyboard type at the beginning of each session. Users were asked to memorize their passwords prior to typing them too. However, it is impossible to ensure that each user did so.

3. **Impostor Data** - Both KDS-1 and KDS-2 coupled users into pairs where one unknowingly acted as the impostor for the other. However, in the real

world, impostor keystroke samples could vary dramatically. Only a study with a larger user base would allow us to automatically generate impostor keystroke data and, subsequently, build reasonable impostor models.

4. **Selection** - When analyzing KDS-2 in Sections 3.9.6-3.9.8, we used a user-password combination as a subject while performing statistical tests. Since each user typed 2 passwords, each user appeared as 2 subjects in the study. Independence between the subjects is required for the Friedman's test. It would be possible to suggest that there is a relationship between these "two" subjects due to the fact they represent the same user. However, our reasoning is that since the passwords being typed are different, the two subjects represented by a single user are likely independent.

5. **Diffusion or imitation** - In KDS-1, subjects were asked to type their username and passwords, followed by the user / password credentials of a few other users. There was no attempt to habituate impostor users typing someone else's credentials, or having them see/hear how the credential's owner types them. Motivated intruders would likely engage in social and training activities that may enhance the probability of a successful intrusion.

3.10.2 Threats to External Validity

1. **Interaction of selection and treatments** - In all datasets, the age of users who participated in collections ranged from low-twenties to late sixties. They were of both genders and with varying typing skills. Very few users listed "hunt and peck" as their typing proficiency, and even amongst them all showed a typing speed of at least 5 characters per second. "Hunt and peck" typists may be a more common demographic in the real world, hence it may be argued that the test population is not representative of a broader population.

2. **Interaction of settings and treatments** - The users who participated in the KDS-1 and KDS-2 data collection were allowed to submit their data at their own convenience. The objective was to conduct the study in a realistic setting. Thus, factors such as posture, external environment, the number of consecutive password entries, and other distractions were not taken into account and may have an impact on the external validity of our findings.

3.11 Chapter Summary

In this chapter, we analyzed two aspects of keystroke dynamics, the first being user habituation. This refers to the change in a user's typing behavior with time as he or she becomes familiarized with the password. We conducted an experimental study to determine the statistical effects of habituation in typing short, simple or long, complex passwords. We demonstrated that there is a statistically significant reduction in typing time over consecutive entries for both short and long passwords. We also showed that the variance of the users' total typing time reduces significantly for both types of passwords. This indicates that, over time, typing behavior gravitates to a user specific but uniform pattern.

The separability of paired users did not show a statistically significant difference in short passwords, likely due to the commonality of the words selected as passwords. For long passwords, we observed that while inter-user separation decreases over time, the intra-user variation also decreases leading to more uniform password typing patterns that are easily distinguishable.

With respect to the effect of habituation on authentication performance, when classifiers were trained using early (noisy) keystroke entries, the performance was significantly worse in comparison with those trained with more uniform data. Such effects of habituation must be taken into account by when deploying keystroke dynamic based authentication systems. When a user is assigned a credential set their typing behavior will show initial high variance until they gain proficiency and convert the patterns to muscle memory. The most effective authentication model in our experience is one that trains on the most recent keystroke entries from each user.

The second aspect of keystroke dynamics analyzed in this chapter is the concept of event sequences. It is based on the flexibility that keyboards offer in typing a complex string using different keystroke sequences. Current keystroke datasets have been recorded by restricting the user to type a single event sequence. Using a locally collected dataset, we performed a series of experiments with keystroke event sequences. These experiments ascertained the effectiveness offered by event sequences in distinguishing users, the correlation between users' typing proficiency and the event sequences, and the effect of habituation on the event sequences used by any user. The results demonstrated that event sequences increase separability between individual users in keystroke dynamics. We provided empirical proof that event sequences are an attribute independent of a user's typing proficiency unlike traditional keystroke dynamics attributes. Lastly, we showed that as users habituate to a given string, the variations in the event sequences decrease significantly.

Chapter 4
Touch Based Recognition

4.1 Motivation

Mobile touch screen devices have become ubiquitous in today's world. Unlike personal computers, their portability allows them to be held in a variety of positions and used while moving around. Furthermore, they come in various sizes and from various manufacturers. Due to their pervasive nature, a continual authentication system based on this modality holds considerable promise. An ideal authentication system would leverage the various usage scenarios for mobile devices (posture, device type, user movement, etc.) to achieve a balance between practicality and effectiveness.

This chapter investigates the effect of three factors on the authentication performance of a touch-based authentication system: user posture, device size and device manufacturer. Each of these has the potential to affect the device usage behavior of a user and thus, the authentication performance. We investigate the effect of these factors by conducting an experimental data collection. This data collection is structured such that each of the three factors (posture, device type and device manufacturer) can be controlled. Using the collected dataset, we conduct a benchmark analysis of various classification algorithms. We then perform statistical tests and performance analysis to determine the effect of each of the three factors on the authentication performance of our touch-based authentication system. The results of these statistical tests will allow us to determine those factors that must be accounted for when developing a user model. For example, if changing the user posture is found to have a significant impact on the classifier performance, it would be advisable to develop separate user models for different postures.

This section of the document is structured as follows: Section 4.2 described the app developed to use in the data collection. Section 4.3 explains the experiment design. Section 4.4 describes the format of the collected data, and the noise removal and feature extraction techniques applied to the dataset. Section 4.5 described the procedure to sample training and testing sets for the purposes of the experiments. Sections 4.6-4.10 answer various questions pertaining to touch dynamics based on an empirical analysis of the dataset. Section 4.13 summarizes the conclusions from the analyses. Section 4.12 critically discusses the assumptions of the dataset and the threats to the validity of the conclusions. Section 5.3 concludes this chapter with a discussion of future directions for research.

Table 20: Characteristics of devices used in the study

ID	Manufacturer	Model	Device Type	Display Size	OS version
T10	Samsung	Tab 2 10”	Tablet	10”	4.1
T7	Samsung	Tab 2 7”	Tablet	7”	4.1
S3	Samsung	S3	Smartphone	4.8”	4.1
EVO	HTC	Evo 4G LTE	Smartphone	4.7”	4.1

4.2 Device selection and design of the data collection app

Four devices were selected for the purposes of this study: two tablets of different display sizes and two smartphones of the same display size but different manufacturers. Their characteristics are described in Table 20. These particular devices were chosen specifically to answer our research questions. In order to minimize the effect of extraneous factors such as installed apps and software aging, the devices were factory reset to their default settings before conducting the data collection. No additional software was installed on the devices except for the data collection app. All selected devices were based on the Android operating system. We chose Android for multiple reasons: According to a developer survey conducted in April-May 2013, Android is used by 71% of mobile-software developers [54]. Furthermore, Android has close to 80% market share in global smartphone shipments in 2013 [27]. Thus, any results we obtain on these devices would be applicable to the larger segment of the mobile device population. Furthermore, Google, Android’s developer, provides a customized Eclipse IDE for Android app development. This allows for rapid development, prototyping and testing of apps [66].

The motivation behind developing a custom app was to create a means to capture the natural horizontal and vertical strokes from the user. The app consisted of a photo matching game where the objective is to find a randomly displayed image from a list of images. The flow of activities when using this app is shown in Fig. 16. When the user begins the game, on Screen 1, he/she attempts to find the displayed photo by scrolling vertically through a list of images. Once the matching photo is found, the user clicks it. At this, Screen 2 is displayed with another image that must be found. On this screen, however, the user must swipe sideways to find the displayed photo. Once the matching photo is found, the user clicks the ‘Back’ button which causes Screen 1 to reappear with another randomly selected image. This pair of activities repeat five times. The game ends after the fifth iteration.

As the user interacts with the app, the touch data is collected in the background. The structure and type of touch data collected is explained later in Section 4.4.

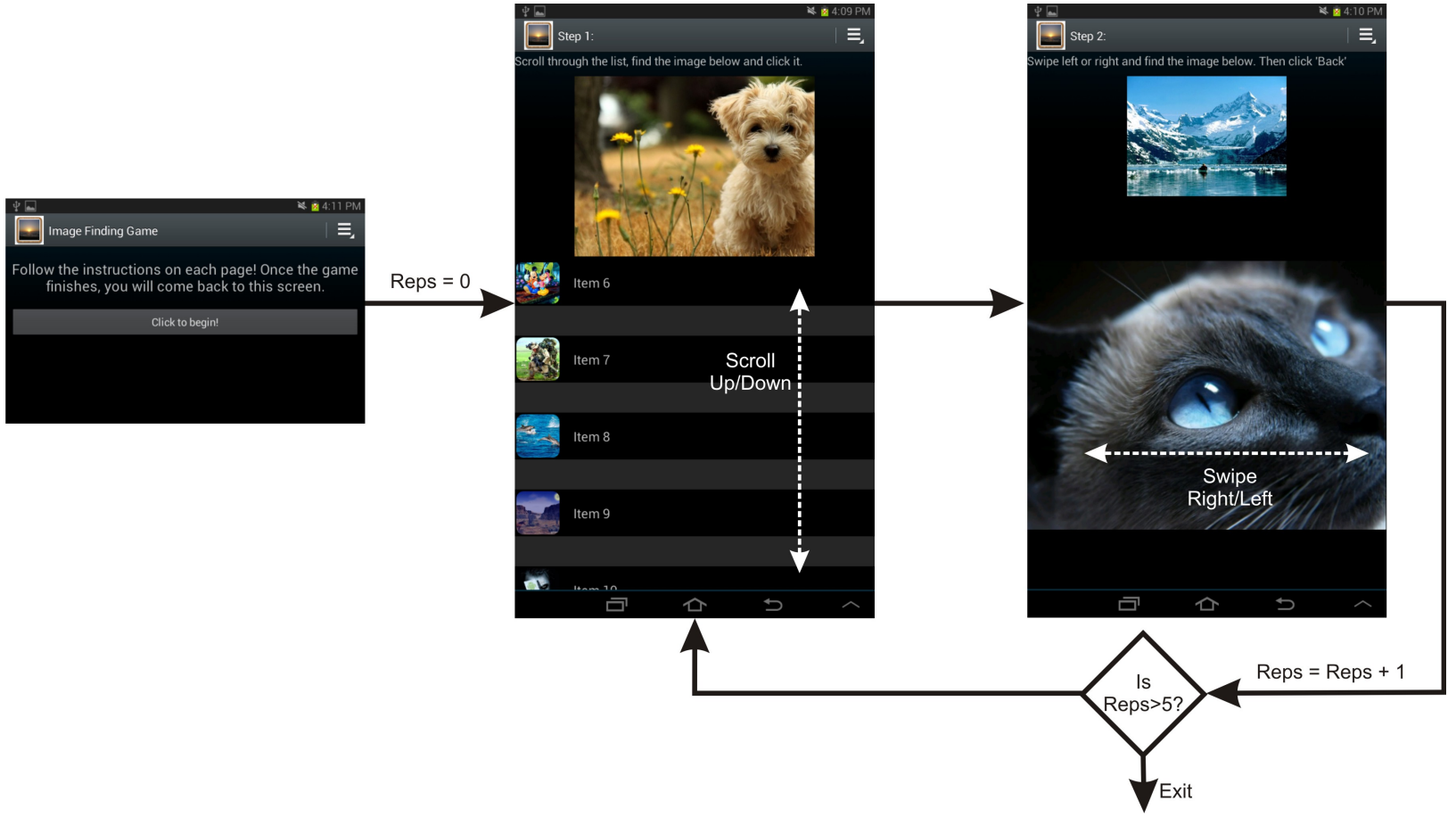


Figure 16: Activity flow in the image-matching app

4.3 Experiment design

As part of the data collection, we recruited 31 test subjects. In a given session, the subject played the game three times on each of the four devices. To reiterate, a game refers to finding a total of ten images in the app (five through scrolling and five through swiping). Each game typically required 3 minutes. One game was played on each of the following postures:

1. P1: Device lying flat on a table in portrait orientation
2. P2: Device held in portrait orientation
3. P3: Device held in landscape orientation

These set of 3 games when played on each of the four devices is referred to as a 'session' in the rest of the work. All test subjects used the same set of devices and were required to be in a seated position. The same set of pictures was used on all devices and throughout the data collection. The data collection was primarily conducted in a laboratory setting. The subjects took part in 8 sessions so that adequate data could be gathered for training and testing user models. The reason for choosing 8 sessions was based on a pilot study performed on 7 users explained later in Section 4.6.

4.4 Noise removal & Feature extraction

The touch data was collected in the app via the standard Android library and the provided methods. The raw touch data gathered by our custom app contains the X/Y coordinates of a touch event, the pressure applied at that spot and the time of the event (measured in nanoseconds). Other collected information included device specific characteristics such as dots per inch, screen resolution in the corresponding orientation, etc.

On the Android OS, the atomic touch data for a gesture consists of a series of touch events. These events are:

1. ACTION_DOWN: A pressed gesture has started, the event contains the initial starting location
2. ACTION_MOVE: A change has happened during a press gesture (between ACTION_DOWN and ACTION_UP). The motion contains the most recent point, as well as any intermediate points since the last down or move event.
3. ACTION_UP: A pressed gesture has finished, the motion contains the final release location as well as any intermediate points since the last down or move event.

Thus, a touch gesture consists of an Action_Down event, followed by a series of Action_Move events and, finally, an Action_Up event. Each event has an X/Y coordinate, pressure and timestamp associated with it.

Before performing feature extraction, we removed certain types of strokes from the dataset. These strokes did not meet the characteristics that we desired to use in our touch based authentication system. These strokes are:

1. Those that changed direction mid-way.
2. Touch presses that consisted only of an Action_Down and an Action_Up event without any Action_Move events in between.

Strokes of all lengths were preserved even if they happened to be short. While it has been shown in literature that very short strokes possess less discriminatory power [24], we chose to keep them in the dataset and let the classifier decide their usefulness. Once the dataset was pre-processed in this manner, we extracted 18 features for each gesture. These are listed in Table 21.

Once the unwanted strokes were removed and the features extracted for each stroke, the dataset was pruned so that all subjects had an equal number of strokes in all device-posture scenarios. This pruning was performed to remove any confounding effect in the results due to some users having a larger number of strokes in their dataset than others.

Table 21: List of features generated and their description

Feature	**Description**
StartX, StartY, StartPressure	Abscissa, ordinate and pressure at the location where the gesture began
StopX, StopY, StopPressure	Abscissa, ordinate and pressure at the location where the gesture ended
StrokeDuration	Duration of stroke (in μs)
Length_EE, Angle_EE	Distance and angle between beginning and end point (in pixels)
Length_Trj	Length of gesture's trajectory
Ratio_TrjLen2EELen	This ratio between Length_EE and Length_Trj. This is a measure of deviation of the gesture from a straight line.
AverageVelocity	The average velocity of the gesture
InterStrokeTime	Delay between successive strokes
MidPressVal	Pressure at the midpoint of the gesture
PairedVel20, PairedVel50, PairedVel80	Average velocity after 20/50/80% of the stroke has been executed
Direction	Primary direction of the stroke (Horizontal/Vertical)

4.5 Stroke sampling for generating training-testing sets, classifier testing procedure and performance metrics

The process of selecting strokes to create training and testing sets and generate performance metrics is illustrated in Fig. 17. The process is divided into seven steps, each described in detail below:
For any given user,

- **Step 1**: For creating the genuine train and test sets, strokes from the user's genuine dataset is sampled such that alternating batches of five strokes across the entire genuine dataset are collated together into groups to form the training and testing sets. Let the total number of such groups formed from the genuine strokes be $2L$. Thus, the entire genuine dataset of the user is sampled evenly for groups of strokes. In doing so, we remove the effects of user habituation that may affect classifier performance if a single contiguous block of strokes from one point in time was employed as the training set
  We sampled groups of strokes instead of single strokes under the reasonable assumption that touch gestures over a short period of time are similar to each other and would thus lead to a more stable user model.

- **Step 2**: For creating the impostor train and test sets, an equal number of groups of strokes are sampled from each impostor such that the total number of sampled groups is equal to $2L$. For each impostor, the groups in his or her dataset are sampled evenly across their entire dataset. This helps account for the effects of habituation that may change that impostor's behavior over time.

- **Step 3**: The train and test sets are created by combining the corresponding genuine and impostor sets. Both the train and test sets contain $2L$ groups of strokes.

  The horizontal and vertical strokes are stored separately. Their train and test sets are constructed separately using Steps 1, 2 and 3. The train and test sets from horizontal and vertical strokes are then concatenated to create the final train and test sets.

- **Step 4**: The classifier is trained on the training set and a user model is generated.

- **Step 5**: Once the user model is constructed, its performance is benchmarked on the testing set. Since the testing set consists of groups of five strokes, each of these groups is tested on the classifier.

- **Step 6**: Each stroke in a group is assigned its own confidence value by the classifier. The confidence value is a measure of the classifier's certainty in labeling the stroke with a particular class label (genuine or impostor).

- **Step 7**: In order to generate performance metrics, a confidence value threshold is set. Based on this threshold, each stroke within the groups is labeled a genuine (if their corresponding value is greater than the threshold) or impostor. The majority class label within the group is then assigned to all gestures in the group. Thus, if the five strokes in a group are labeled as genuine by the classifier with confidence values 0.30, 0.79, 0.65, 0.91, and 1.00, then based on a confidence value threshold of 0.60, all strokes in the group will be labeled as genuine. Once the entire testing set has been classified in this manner, the False Accept Rate and False Reject Rate are calculated:
  $False\,Accept\,Rate = \frac{\#\,of\,impostor\,instances\,classified\,as\,genuine}{Total\,\#\,of\,impostor\,instances}$
  $False\,Reject\,Rate = \frac{\#\,of\,genuine\,instances\,classified\,as\,impostor}{Total\,\#\,of\,genuine\,instances}$
  The confidence value threshold can be lowered to make the classifier user-friendly (and less trustworthy). It can also be increased to make the classifier highly restrictive (and more trustworthy). For example, in a Random Forest classifier, the threshold parameter refers to the number of trees in the ensemble that voted for a particular class. At a certain threshold value, the FAR is equal to the FRR. This FAR/FRR value is called the Equal Error Rate (EER). The EER serves as a single-valued performance metric for the classifier. In our experiments, the EER was chosen as the default performance metric unless otherwise noted.
  When the classification threshold is varied in steps across its range, and the FAR and True Positive Rate (1-FRR) are plotted at each data point, the resulting graph is called the Receiver Operating Characteristics (ROC) curve. The ROC curve for a classifier, shown in Fig. 18, illustrates its performance as a trade off between selectivity and sensitivity.

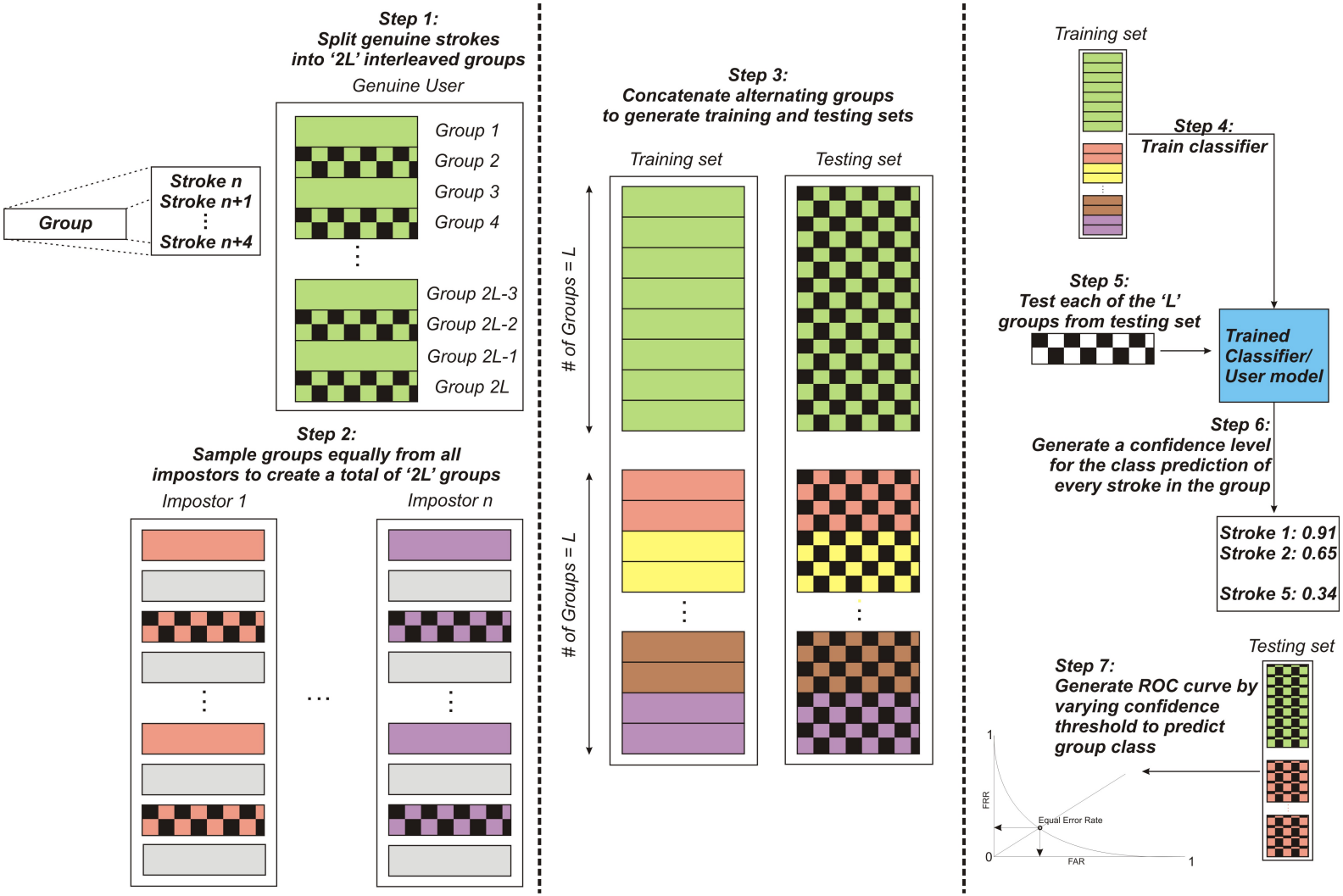


Figure 17: The procedure for extracting training and testing sets from the touch data, training the classifier, and generating performance metrics

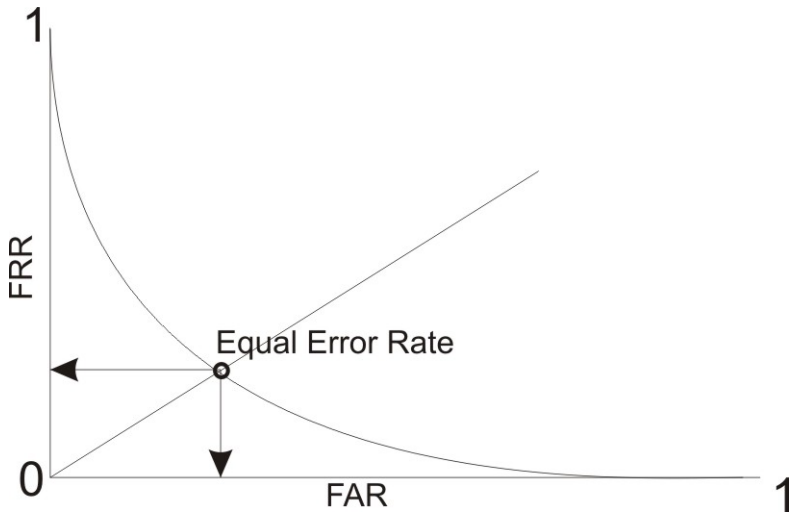


Figure 18: A sample ROC curve illustrating the trade-off between selectivity and sensitivity of a classifier [30]

4.6 Variation in classifier performance with change in size of training set

Before the full fledged data collection, we conducted a pilot study over 7 users on the T7 tabet.. This preliminary study was conducted to determine the amount of training data necessary to create an adequate user model. This in turn would help us define the number of sessions we needed to conduct per user in the actual data collection. To answer this question, each of the 7 users took part in 10 sessions. We reiterate that a session refers to playing 3 games on each of the four devices with each of the 3 games played in a different posture. The touch data was then pre-processed as described in Section 4.4. In order to determine the optimal amount of training data needed to create an adequate user model, we performed the following steps:

1. The user data was split equally into train and test data. The training and testing data was extracted as described in Section 4.5.
2. 10 train sets of varying sizes were created by varying the amount of training data from 10%-100% of the complete training data (in steps of 10%).
3. Each of the 10 training sets was used to create a user model based on the Random Forest classifier.
4. For every user, the 10 user models were tested against the testing set (which is of a constant size).
5. The mean EER for all users at every training set size was calculated.

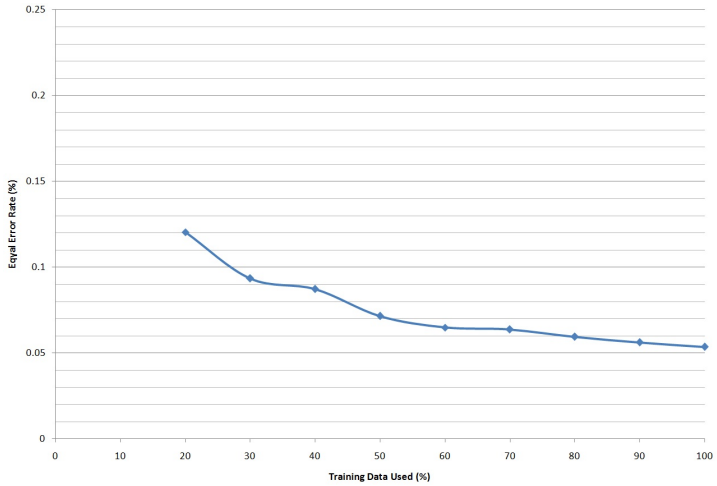


Figure 19: Variation in EER as the size of the training set size is increased

The variation in mean equal error rate with training set size is shown in Fig. 19. Note that once the training set size reaches 80% (data from 4 sessions), the EER of the classifier does not decrease by more than 1% upon adding more training data. Based on this analysis, we decided to collect data over 8 sessions to gather sufficient data for training and testing sets in the full-fledged data collection.

4.7 Determining the best performing classification algorithm

The next objective was to determine the best classification algorithm and use it as the basis for further experiments. We tested five algorithms based on a benchmark study performed by Serwadda et al [67]. The authors here reported that Logistic Regression, SVM, Random Forest, Naive Bayes, and Multi-layer Perceptron were the best performing algorithms. To find the best classifier, we followed the same steps as described in Section 4.6. However, for every user model in the previous example, we developed five in this experiment, where each user model was created using one of the five benchmark classification algorithms.

The variation in EER for each classifier for this experiment is illustrated in Fig. 20. The mean EER and its standard deviation for each classifier for all users are given in Table 22. It is evident that the Random Forest classifier performed the best amongst all algorithms. When using only 10% of the total training set, it records an EER of 17.6%. As the training set size is increased, the performance increases to provide a benchmark EER of 5.4%. Comparatively, Multi-layer Perceptron was the worst performing classifier. Table 22 also indicates that Random Forest provides the most reliable classification. This is

evident from the smaller change in performance as the training set size is increased. Furthermore, it exhibits a lower standard deviation as compared to the other classifiers, indicating that it provides more reliable performance across a spectrum of users. Based on these results, we decided to use Random Forest to generate user models for the remainder of the experiments.

Interestingly, the classification algorithms exhibited oscillation in the Equal Error Rate as the training set size was increased. This behavior was most pronounced for the Naive Bayes and Multi-layer Perceptron algorithms. It may be suggested that this behavior is due to difference in user behavior over time. Thus, a shorter training set would reflect the user in his/her earlier stages of familiarity with the app. However, we would like to remind the user that the entire training set was sampled evenly in order to generate a smaller training set. This was done to remove the effects of variation in user behavior over time. The reason for this fluctuation in performance requires further investigation.

Using the same experimental setup, we evaluated the effectiveness of the features used to construct the user model. An Information Gain evaluator was used to determine the discriminatory power of a feature and the attributes were ranked according to their individual evaluations. The results of this test are shown in Table 23. A key point to note from the analysis is that, within this dataset, the pressure data possesses negligible discriminatory power. The most effective features are based on gesture coordinates, length and speed. This may be because the pressure readings from touchscreen devices are not accurate enough to be used for identification purposes. A key reason for this is the manner in which pressure data is gathered on touch screen devices. This has been explained in Section 2.2.9. To reiterate, the pressure is interpreted via the size of contact area between the finger tip and the screen, and not via a dedicated pressure sensor. It is probably due to these reasons that this indirectly sensed pressure data does not have sufficient discriminatory power.

4.8 Experiment 1: Given a posture, does device size affect the user profile w.r.t. classifier performance?

With the rapid rise in popularity of mobile devices, it is a common trend that users have multiple touch screen devices. In such cases, a highly desirable characteristic of a touch based authentication system is to use the collate data from multiple devices and use it across different devices. This will either allow a user profile developed on one device to be ported to another device. Furthermore, it may even be possible to merge two profiles from different devices together to rapidly create a user model or a more robust authentication system. Based on this motivation, we decided to study whether the device size has any effect on the user profile.

In this experiment we restricted the user posture in order to remove a confounding variable. We argue that in a practical system, it is quite possible to

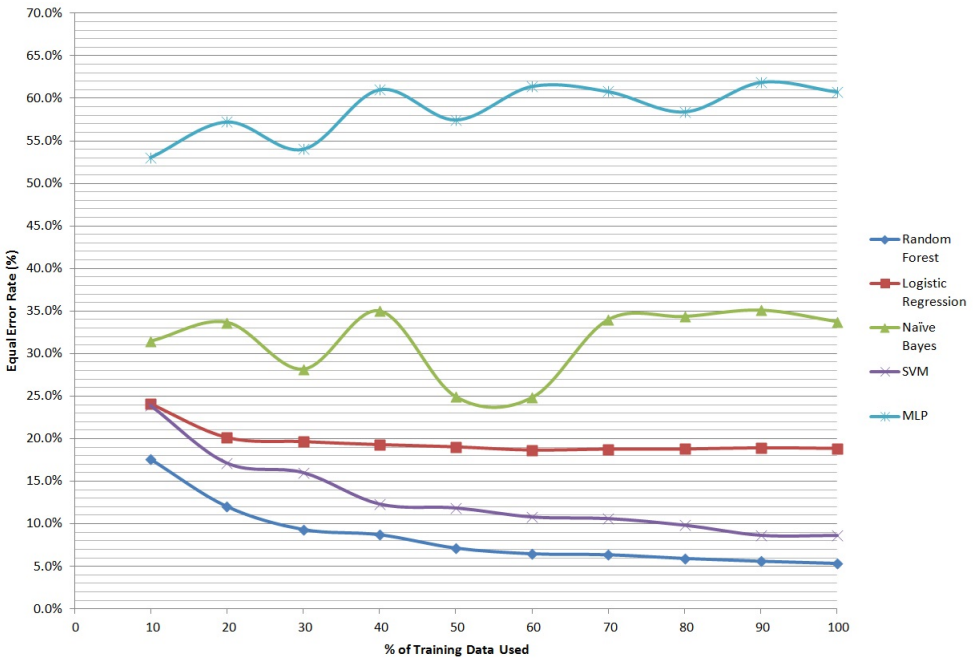


Figure 20: Performance of various classifiers with variation in training set size. The Random Forest classifier performed the best followed by SVM.

Table 22: Mean classifier performance over all 31 users as the training set size is varied. The values in parentheses show the standard deviation.

Training Data Used %	Random Forest	SVM	Logistic Regression	Naïve Bayes	MLP
10	17.6% - (7.8%)	24.1% - (8.6%)	31.4% - (15.0%)	23.9% - (7.8%)	53.0% - (17.6%)
20	12.0% - (7.5%)	20.1% - (8.0%)	33.7% - (18.7%)	17.1% - (7.5%)	57.2% - (18.4%)
30	9.3% - (6.8%)	19.6% - (8.1%)	28.2% - (12.8%)	16.0% - (6.8%)	54.0% - (14.7%)
40	8.7% - (5.8%)	19.3% - (7.6%)	35.0% - (16.8%)	12.3% - (5.8%)	61.0% - (12.2%)
50	7.2% - (5.8%)	19.0% - (7.9%)	25.0% - (11.8%)	11.9% - (5.8%)	57.5% - (13.0%)
60	6.5% - (5.9%)	18.7% - (7.7%)	24.9% - (11.0%)	10.8% - (5.9%)	61.4% - (12.6%)
70	6.4% - (4.9%)	18.8% - (8.0%)	34.0% - (14.5%)	10.6% - (4.9%)	60.8% - (15.5%)
80	5.9% - (4.6%)	18.8% - (7.9%)	34.3% - (14.0%)	9.8% - (4.6%)	58.4% - (12.3%)
90	5.6% - (4.2%)	18.9% - (8.3%)	35.1% - (12.8%)	8.6% - (4.2%)	61.9% - (11.8%)
100	5.4% - (4.1%)	18.8% - (7.8%)	33.7% - (12.7%)	8.6% - (4.1%)	60.8% - (13.1%)

Table 23: Effectiveness of features used to generate user models

Rank	Info. Gain	Feature	Rank	Info. Gain	Feature
1	0.4072	StartX	11	0.061	StartPressure
2	0.3654	StopX	12	0.051	PairedVel20
3	0.1891	StopY	13	0.0496	PairedVel80
4	0.1301	MidPressVal	14	0.0182	StopPressure
5	0.109	AverageVelocity	15	0	Ratio_TrjLen2EELen
6	0.0923	StartY	16	0	StrokeDuration
7	0.092	Length_Trj	17	0	Angle_EE
8	0.0906	Length_EE	18	0	Direction
9	0.0769	PairedVel50			
10	0.0706	InterStrokeTime			

develop user profile based on user postures since the posture can be detected using the device's inbuilt accelerometer and gyrometer readings. In order to study the effect of device size, we performed a statistical comparison using the following steps. The test setup is illustrated in Table 24:

1. From the touch data for a given user and posture, 3 pairs of training and testing sets were extracted using the approach described in Section 4.5. Each set contained the same number of strokes and was constructed using the touch data from one of three devices (T10: 10" tablet, T7: 7" tablet and S3: 4.7" smartphone). These three devices, described in Section 4.2 are all made by Samsung. We excluded the HTC Evo from this experiment in order to keep the device manufacturer same for all devices under test. The training sets for a user are labeled $Train^{T10}$, $Train^{T7}$, and $Train^{S3}$. The testings set are labeled $Test_{T10}$, $Test_{T7}$, and $Test_{S3}$.

2. Three user models were constructed using $Train^{T10}$, $Train^{T7}$, and $Train^{S3}$ with the Random Forest classifier.

3. Each model was then tested on each of the three testing sets and the Equal Error Rate was calculated in every instance. The EER when training on Device X and testing on Device Y for User i is labeled $EER\_i_Y^X$.
   For example, in order to determine the effect of change in device size on User x's model from the T10 device: We restrict the posture and construct the user model for User x using the train set for User x from T10 ($Train^{T10}$) and then benchmark on the test sets for User x from T10, T7, and S3 ($Test_{10}$, $Test_{T7}$, and $Test_{S3}$). This provides the EERs $EER\_x_{T10}^{T10}$, $EER\_x_{T7}^{T10}$ and $EER\_x_{S3}^{T10}$ for that posture.
   When this procedure is performed over all users, it provides a distribution of EER scores as shown in Table 24.

4. To test the effect of device size on authentication performance of a user model based on $Train^X$, the EER distributions EER^X_{T10}, EER^X_{T7}, EER^X_{S3} are compared to each other using a one-sided Student's t-test with Holm-Bonferroni correction.

4.8.1 Student's t-test

The student's t-test is a method of testing hypothesis about the mean of a small sample drawn from a normally distributed population when the population standard deviation is unknown. The formulated null hypothesis states that there is no effective difference between the observed sample mean and the hypothesized or stated population mean, i.e., that any measured difference is due only to chance.

In general, a t-test may be either two-sided (also termed two-tailed), stating simply that the means are not equivalent. The t-test may also be one-sided, specifying whether the observed mean is larger or smaller than the hypothesized mean. The test statistic t is then calculated. If the observed t-statistic is more extreme than the critical value determined by the appropriate reference distribution, the null hypothesis is rejected. The appropriate reference distribution for the t-statistic is the t distribution. The critical value depends on the significance level (alpha level) of the test. This is the probability of erroneously rejecting the null hypothesis [22].

4.8.2 Holms-Bonferroni Correction

Since this experiment (and the following ones) employs multiple statistical tests, it is possible to reject the null hypothesis even if it is correct. This is because as the number of statistical tests increases, the probability of rejecting a null hypothesis at a given alpha level also increases resulting in the inflation of the alpha level. In order to compensate for this problem, we use the Bonferroni-Holm correction for multiple comparisons. This is a sequentially rejective version of the simple Bonferroni correction for multiple comparisons and strongly controls the family-wise error rate at level alpha. It works as follows:

1. All p-values are sorted in order of smallest to largest, where m is the number p-values.

2. If the 1^{st} p-value is greater than or equal to α/m, the procedure is stopped and no p-values are significant. Else, continue.

3. The 1^{st} p-value is declared significant and now the second p-value is compared to $\alpha/(m-1)$.

4. If the 2^{nd} p-value is greater than or equal to $\alpha/(m-1)$, the procedure is stopped and no further p-values are significant.

5. This process continues for all p-values.

4.8.3 Results

Table 25 shows the statistical results of the experiments. Each statistical test is a two-tailed test, i.e. the p-value indicates whether the two distributions of EERs are equal. In our experiments, we required a significance level $\alpha = 0.05$ i.e, the p-value must be lower than 0.01 to reject the NULL hypothesis, i.e. to conclude that the distributions are not equal. The conclusions from the experiment are as follows:

1. As Table 25 shows, the performance of a user model trained on any device is very different from when that same profile is used on another device.

2. This effect is explained in Table 26 where the EER on each device for this experiment are shown. When a user model trained on Device X is used on a test set from Device Y, the performance of the model is equivalent to a coin toss, i.e. the classifier is unable to distinguish between a genuine and impostor user. This shows that based on the current state-of-the-art techniques for feature generation, it is not possible to use user models across devices.

3. The abysmal performance of Train-on-X-Test-on-Y classifiers is somewhat less for the T7 and S3 pair. This seems to suggest that similar device sizes do lead to similar profiles. To evaluate this further, we carried out this same experiment on two devices of the same size but from different manufacturers. This experiment is explained later in Section 4.10.

4. We note that when training and testing on data from the same device, the 10" tablet offers the best authentication performance with EER as low as 3.8%. Comparatively, the S3 smartphone has the worst performance (as low as 8.8%). This suggests that touch based authentication is more effective on devices with a larger touch screen area. This is possibly because the large screen area provides users with greater freedom to select certain screen areas to execute gestures.

5. In all cases where the model is trained and tested on the same device and posture, the classifier performance is either better or equivalent to the benchmark performances in literature. This has also been achieved by using fewer features (18) than other studies in literature. For example, Frank et al. [24] used 27 features, Feng et al. [23] used 53 features and Serwadda et al. [67] used 28 features.

Table 24: Test setup to measure the effect of device on authentication accuracy (given a specific posture).

User	$Train^{T10}$			$Train^{T7}$			$Train^{S3}$		
	$Test_{T10}$	$Test_{T7}$	$Test_{S3}$	$Test_{T10}$	$Test_{T7}$	$Test_{S3}$	$Test_{T10}$	$Test_{T7}$	$Test_{S3}$
1	$EER\_1^{T10}_{T10}$	$EER\_1^{T10}_{T7}$	$EER\_1^{T10}_{S3}$	$EER\_1^{T7}_{T10}$	$EER\_1^{T7}_{T7}$	$EER\_1^{T7}_{S3}$	$EER\_1^{S3}_{T10}$	$EER\_1^{S3}_{T7}$	$EER\_1^{S3}_{S3}$
...	...	...	...	...	...	...	...	...	...
n	$EER\_n^{T10}_{T10}$	$EER\_n^{T10}_{T7}$	$EER\_n^{T10}_{S3}$	$EER\_n^{T7}_{T10}$	$EER\_n^{T7}_{T7}$	$EER\_n^{T7}_{S3}$	$EER\_n^{S3}_{T10}$	$EER\_n^{S3}_{T7}$	$EER\_n^{S3}_{S3}$

Table 25: List of p-values of statistical tests conducted to determine the effect of device-to-device variation when the posture is controlled. Values in red indicate a rejection of the NULL hypothesis. The data shows that there is a very strong difference between touch profiles built using the three devices in any posture.

Train Device:	$Train^{T10}$		$Train^{T7}$		$Train^{S3}$	
EERs compared:	$Test_{T10}$-$Test_{T7}$	$Test_{T10}$-$Test_{S3}$	$Test_{T7}$-$Test_{T10}$	$Test_{T7}$-$Test_{S3}$	$Test_{S3}$-$Test_{T10}$	$Test_{S3}$-$Test_{T7}$
Posture 1	7.94E-18	7.03E-20	1.32E-16	1.00E-16	4.37E-16	1.80E-13
Posture 2	3.08E-15	4.92E-22	3.08E-15	2.96E-12	6.31E-18	2.30E-14
Posture 3	1.05E-16	2.66E-17	4.59E-16	2.82E-13	2.83E-15	2.89E-14

Table 26: Mean EER (over all users) when the model is trained on one device and tested on another device (while the posture is controlled)

Posture	$Train^{T10}$			$Train^{T7}$			$Train^{S3}$		
	$Test_{T10}$	$Test_{T7}$	$Test_{S3}$	$Test_{T10}$	$Test_{T7}$	$Test_{S3}$	$Test_{T10}$	$Test_{T7}$	$Test_{S3}$
Posture 1	5.36%	56.58%	58.69%	58.75%	6.64%	33.97%	55.02%	29.52%	8.81%
Posture 2	5.16%	53.01%	56.75%	52.33%	5.49%	34.30%	54.45%	27.51%	7.14%
Posture 3	3.80%	56.05%	54.37%	55.03%	5.42%	32.19%	51.82%	28.89%	6.60%

4.9 Experiment 2: Given a device, does posture affect the user profile w.r.t. classifier performance?

One of the major advantanges of portable devices is that users are not confined to a certain location or fixed posture to use them. A robust touch-based authentication system must be able to authenticate users even when the user posture changes. This experiment was performed to determine the effects of changes in user posture on authentication performance. The experiment setup is similar to the previous experiment described in Section 4.8. However, in this case, the device type is controlled while the posture type is varied to create training and testing sets. The test setup is illustrated in Table 27 and the experiment is setup as follows:

1. From the touch data for a given user and device, 3 pairs of training and testing sets were extracted using the interleaved approach. The training and testing sets contained the same number of strokes.

2. The training/testing sets were extracted using the touch data from three postures (P1: device on table, P2: device held in portrait orientation and P3: device held in landscape orientation). The training sets for a user are labeled $Train^{P1}$, $Train^{P2}$, and $Train^{P3}$. The testings set are labeled $Test_{P1}$, $Test_{P2}$, and $Test_{P3}$.

3. Three user models were constructed using $Train^{P1}$, $Train^{P2}$, and $Train^{P3}$.

4. Each model was then tested on each of the three testing sets and the equal error rate was calculated in every instance. The EER when training on Posture X and testing on Posture Y for User i is labeled $EER\_i_Y^X$.
   For example, in order to determine the effect of change in posture on User x's model based on Posture 1: We restrict the device type and construct the user model for User x using the train set for User x from Posture 1 ($Train^{P1}$) and then benchmark on the test sets for User x from Postures 1, 2 and 3 ($Test_{P1}$, $Test_{P2}$, and $Test_{P3}$). This provides the EERs $EER\_x_{P1}^{P1}$, $EER\_x_{P2}^{P1}$ and $EER\_x_{P3}^{P1}$ for that posture.
   When this procedure is performed over all users, it provides a distribution of EER scores as shown in Table 27.

5. To test the effect of posture on authentication performance of a user model based on Posture X, the EER distributions EER_{P1}^X, EER_{P2}^X, EER_{P3}^X are compared to each other using a one-sided Student's t-test with Holm-Bonferroni correction.

4.9.1 Results

Table 28 shows the statistical results of the experiments. Each statistical test is a two-tailed test, i.e. the p-value indicates whether the two distributions are

equal. If a significance level $\alpha = 0.05$ is assumed, the p-value must be lower than 0.01 to reject the NULL hypothesis, i.e. to conclude that the distributions are not equal.

1. As Table 28 shows, the performance of a user model trained on any posture is very different from when that same profile is used on another posture. However, the p-value is much less than the previous experiment. This indicates that the difference caused by posture, although significant, is much less than that caused by device to device variation.

2. This effect is further explained in Table 29 where the EER for each posture for this experiment are shown. When a user model trained on Posture X is used on a test set from Posture Y, the performance of the model drops by 3-4 times on the T10 and up to 10 fold on the S3. Comparing between Posture 1 and Posture 2, the difference in performance is at least 4 fold although both postures required a portrait orientation (the only difference between the two postures is that the device was either laid flat on the table or held in one hand).

3. The difference in performance, however, is more pronounced when the model is trained on a landscape orientation and tested on a portrait orientation (or vice-versa). This is to be unexpected since the user interface changes drastically between these two orientations. In order to mitigate this problem, a possible area of future research would be to refer to gesture coordinates in terms of its proximity to certain UI elements, and not in absolute X and Y coordinates.

4. Lastly, a common trend in all scenarios was that Posture 1 (device lying flat on table) had the worst performance. Contrastingly, Posture 3 (device held in landscape mode) offered the best performance. It seems that the users touch profile becomes more specific when the device is held in the hand. Based on this analysis, we believe that utilizing the 3-dimensional orientation data of a device may offer an even better user-specific profile when it is held in the hand(s).

Table 27: Test setup to measure the effect of posture on authentication accuracy (given a specific device)

User	$Train^{P1}$			$Train^{P2}$			$Train^{P3}$		
	$Test_{P1}$	$Test_{P2}$	$Test_{P3}$	$Test_{P1}$	$Test_{P2}$	$Test_{P3}$	$Test_{P1}$	$Test_{P2}$	$Test_{P3}$
1	$EER\_1_{P1}^{P1}$	$EER\_1_{P2}^{P1}$	$EER\_1_{P3}^{P1}$	$EER\_1_{P1}^{P2}$	$EER\_1_{P2}^{P2}$	$EER\_1_{P3}^{P2}$	$EER\_1_{P1}^{P3}$	$EER\_1_{P2}^{P3}$	$EER\_1_{P3}^{P3}$
...	...	...	...	...	...	...	...	...	...
n	$EER\_n_{P1}^{P1}$	$EER\_n_{P2}^{P1}$	$EER\_n_{P3}^{P1}$	$EER\_n_{P1}^{P2}$	$EER\_n_{P2}^{P2}$	$EER\_n_{P3}^{P2}$	$EER\_n_{P1}^{P3}$	$EER\_n_{P2}^{P3}$	$EER\_n_{P3}^{P3}$

Table 28: List of p-values of statistical tests conducted to determine the effect of posture variation when the device type is controlled. Values in red indicate a rejection of the NULL hypothesis. The data shows that there is a strong difference between touch profiles based on what posture the user is in. However, it is much less than the differences in touch profile caused by device-to-device variation.

Train Posture:	$Train^{P1}$		$Train^{P2}$		$Train^{P3}$	
EERs compared:	$Test_{P1}$-$Test_{P2}$	$Test_{P1}$-$Test_{P3}$	$Test_{P2}$-$Test_{P1}$	$Test_{P2}$-$Test_{P3}$	$Test_{P3}$-$Test_{P1}$	$Test_{P3}$-$Test_{P2}$
T10	4.66E-06	1.24E-08	2.71E-08	1.24E-08	1.40E-10	3.03E-11
T7	8.52E-08	4.44E-10	1.23E-08	1.50E-09	2.52E-12	1.08E-10
S3	1.62E-07	8.14E-12	9.78E-10	2.76E-12	2.81E-14	6.64E-12

Table 29: Mean EER (over all users) when the model is trained on one posture and tested on another posture (while the device type is controlled)

Device	$Train^{P1}$			$Train^{P2}$			$Train^{P3}$		
	$Test_{P1}$	$Test_{P2}$	$Test_{P3}$	$Test_{P1}$	$Test_{P2}$	$Test_{P3}$	$Test_{P1}$	$Test_{P2}$	$Test_{P3}$
T10	5.36%	21.11%	36.83%	20.78%	5.16%	23.94%	41.84%	30.14%	3.80%
T7	6.64%	23.31%	34.00%	21.37%	5.49%	26.31%	35.75%	27.45%	5.42%
S3	8.81%	19.67%	33.65%	21.64%	7.14%	37.02%	39.86%	37.05%	6.60%

4.10 Experiment 3: Does the type of manufacturer affect a user's touch profile w.r.t. authentication performance?

Generally, manufacturers source parts from Original Equipment Manufacturers (OEMs). For example, the processor ships from one company, the touchscreen digitizer from another, etc. The touch screen brand may have an effect on the screen sensitivity and response time of the device. Additionally, the effects on performance due to hardware and software specifications cannot be easily discerned. The above experiments show that the posture both posture and device size have a significant effect on the a touch dynamics based authentication system. We also noted that the T7 and S3, being closer in size, displayed better performance when their user models were cross-tested. Thus, our next step was to determine the effect of the device manufacturer and if a user model is portable between similar sized devices.

To do so, the experiment was setup exactly as in Section 4.8. However, only two devices were used in this experiment: the HTC EVO and the Samsung S3. This is because both devices have the same display size (4.7"). The test setup is illustrated in Table 30. To reiterate:

1. From the touch data for a given user and posture, 2 pairs of training and testing sets were extracted using the interleaved approach. The training and testing sets contained the same number of strokes.

2. The training/testing sets were extracted using the touch data from two devices (S3 and EVO). The training sets for a user are labeled $Train^{S3}$, and $Train^{EVO}$. The testings set are labeled $Test_{S3}$, and $Test_{EVO}$.

3. Two user models were constructed using $Train^{S3}$, and $Train^{EVO}$.

4. Each model was then tested on each of the two testing sets and the equal error rate was calculated in every instance. The EER when training on Device X and testing on Device Y for User i is labeled $EER\_i_Y^X$.
   For example, in order to determine the effect of using the EVO model on the S3 for User x: We restrict the posture and construct the user model for User x using the train set for User x from EVO ($Train^{EVO}$) and then benchmark on the test sets for User x from EVO and S3 ($Test_{EVO}$ and $Test_{S3}$). This provides the EERs $EER\_x_{EVO}^{T10}$ and $EER\_x_{S3}^{T10}$ for that posture.
   When this procedure is performed over all users, it provides a distribution of EER scores as shown in Table 30.

5. To test the effect of device size on authentication performance of a user model based on Device X, the EER distributions EER_{S3}^X and EER_{EVO}^X are compared to each other using Student's t-test with Holm-Bonferroni correction.

4.10.1 Results

Table 31 shows the statistical results of the experiments. Each statistical test is a two-tailed test, i.e. the p-value indicates whether the two EER distributions are equal. A significance level $\alpha = 0.05$ is assumed. Therefore, the p-value must be lower than 0.01 to reject the NULL hypothesis, i.e. to conclude that the EER distributions are not equal.

1. As Table 31 shows, the performance of a user model trained on one manufacturer is significantly different from when that same profile is used on another manufacturer's device.

2. This effect is explained in Table 32 where the mean EERs on each device for this experiment are shown. When a user model trained on Device X is used on a test set from Device Y, the error rate roughly doubles in each case. This happens even though the device size and posture is the same. It should be noted that the drop in performance is far less compared to Experiment 1 where a different sized device from the same manufacturer was used.

3. Based on the observations in Section 4.8 and from this experiment, it is evident that a similar device size results in a similar user profile. In this experiment, since the manufacturers of the devices are different, this may have led to a larger drop in EER when the user model is ported over. The exact reasons for variation in performance forms an interesting area for further research.

4. When training and testing on data from the same device, the S3 offers better authentication performance compared to the EVO. This may be due to the hardware and/or software configuration of the devices. However, determining the exact play of variables that cause this performance difference is infeasible because a computer is a combination of hardware and software components that function together to create its unique behavior. Thus, testing each component independently is not possible.

5. We note that the trend from Experiment 2 is exhibited here too: Posture 1 provides the worst authentication performance, while Posture 3 provides the best performance for the EVO and S3 smartphones.

Table 30: Test setup to measure the effect of manufacturer on authentication accuracy

User	$Train^{S3}$		$Train^{EVO}$	
	$Test_{S3}$	$Test_{EVO}$	$Test_{S3}$	$Test_{EVO}$
1	$EER\_1^{S3}_{S3}$	$EER\_1^{S3}_{EVO}$	$EER\_1^{EVO}_{S3}$	$EER\_1^{EVO}_{EVO}$
...	...	...	...	...
n	$EER\_n^{S3}_{S3}$	$EER\_n^{S3}_{EVO}$	$EER\_n^{EVO}_{S3}$	$EER\_n^{EVO}_{EVO}$

Table 31: List of p-values of statistical tests conducted to determine the effect of manufacturer variation when the posture is controlled. Values in red indicate a rejection of the NULL hypothesis. The data shows that there is a strong difference between touch profiles based on the phone manufacturer.

Train Device:	$Train^{S3}$	$Train^{EVO}$
EERs compared:	$Test_{S3}$-$Test_{EVO}$	$Test_{EVO}$-$Test_{S3}$
Posture 1	2.51E-08	1.72E-10
Posture 2	7.79E-08	2.01E-11
Posture 3	2.86E-10	2.84E-09

Table 32: Mean EER (over all users) when the model is trained on one device and tested on the other manufacturer's device

	Posture 1		**Posture 2**		**Posture 3**	
	$Test_{S3}$	$Test_{EVO}$	$Test_{S3}$	$Test_{EVO}$	$Test_{S3}$	$Test_{EVO}$
$Train^{S3}$	8.8%	16.5%	7.1%	14.6%	6.6%	14.0%
$Train^{EVO}$	17.5%	8.1%	17.4%	6.7%	16.7%	5.5%

4.11 Time to authentication

A continual authentication system actively monitors user behavior for impostors. Due to the unconstrained nature of this authentication mechanism, there may be a lull in user activity, preventing the system from rapidly determining the user's identity. Due to this, a continual authentication system requires more time to authenticate a user as compared to static authentication measures. Thus, the time to authentication is an important performance metric for continual authentication systems.

Table 33 illustrates the procedure to calculate the time to authentication for the touch-based authentication system proposed in Section 4.5. Note that our system requires 5 strokes to generate a decision. Based on this information and statistics from the collected dataset, our calculations show that given any device-posture combination, our authentication system is able to determine the user's identity in 18 seconds. If the user model native to that device-posture combination is used for authentication, then our proposed system achieves the performance summarized in Table 34 within a span of 18 seconds.

Please note that this performance and time-to-authenticate is based upon the frequency with which users submitted strokes during the data collection. Due to the focused nature of the data collection where the users interacted with the device without a pause, 18 seconds is possibly the shortest possible time-to-authentication for our system. In practical situations, the user may be less focused and may execute gestures with reduced frequency leading to a longer time to authentication. This problem can be offset by reducing the number of strokes used to make a decision. This, however, would impact the system performance.

It is highly likely that the system's authentication performance is proportional to the number of strokes used to make a decision about the user's identity. For example, using ten strokes to authenticate a person will probably reduce the EER further, but increase the time-to-authentication. This trade-off between strokes-to-decsion and time-to-decision is an interesting research topic that merits further investigation.

4.12 Critical discussion

4.12.1 Experimental setup

During the data collection, a maximum of four test subject submitted data concurrently. In all cases, the order in which the devices were used was randomized. Subjects did not sit near each other so they could not imitate each other. The same set of devices was used for all test subjects to minimize the effect of device variation. The devices were reset to their factory default to remove the effect of software slowdown. The data submission process typically took 1-2 weeks. This format was used in order to incorporate information on user acclimatization and the effect of device usage over multiple sessions.

Table 33: Statistics on an average user's in the collected dataset and expected time to authentication. Please note that the provided statistics are for any given device-posture combination

Total number of sessions	8
Avg. time taken to complete a session	$9\,mins.$
Total strokes recorded	1252
# of strokes executed/min.	$\frac{1252}{8\times9} = 17$
Strokes reqd. for authentication	5
Time to authentication	$\frac{5}{17} \times 60 =\sim 18\,seconds$
% of strokes lost in noise removal	13%
Strokes/user in pruned dataset	1090
Horizontal:Vertical strokes	5 : 1
# of horizontal strokes	908
# of vertical strokes	182

Table 34: Mean EER for our touch-based authentication system when using the native device-posture user models. (**Posture 1**: Device on table; **Posture 2**: Device held in portrait mode; **Posture 3**: Device held in landscape mode).

Device	**Posture**		
	P1	*P2*	*P3*
T10	5.36%	5.16%	3.80%
T7	6.64%	5.49%	5.42%
S3	8.81%	7.14%	6.60%

4.12.2 Controlling confounding factors

The experimental setup allowed us to control certain confounding factors:

1. Variations due to the type of device used as the manufacturer, model, touch screen technology (capacitive, resistive), screen size, screen resolution, aspect ratio, and touch screen sampling rate. This was controlled by using the same set of devices for all test subjects.

2. Variations caused by user movement during device interaction. This was controlled by conducting the data collection in a laboratory environment with the same desk and table for all test subjects.

3. Variations in user behavior due to extended device use. The users were expected to complete one session per sitting. The reason for this was to extract a user behavior over a period of time. At the same time, users would have learnt to use the app as they familiarized with it. However, this behavior was part of the study and is not considered a confounding factor.

Some confounding factors in this study are:

1. **Repeated testing** - The users habituated to the app over time. However, This was done intentionally in order to incorporate information on user habituation into the data. Hence, it is not a threat to its validity.

2. **Impostor Data** - Due to the limited number of test subjects in our study, we cannot generate a sufficiently varied impostor model. In the real world, impostor touch data could vary dramatically. Only a study with a larger user base would allow us to build reasonable impostor models.

3. **Targeted attack verses random attack** - While other test subjects function as impostors for a given subject, they did not try to mimic the touch behavior of the user. A more sophisticated impostor would watch the user for some time and mimic his/her behavior. However, as a counterpoint, it seems improbable that an impostor could learn to mimic the amount of pressure and of acceleration at different parts of the stroke just by watching the genuine user.

4. **Interaction of selection and treatments** - All test subjects in this data collection were students with ages ranging from 16-30. This demographic is not representative of the general population. Furthermore, only one subject indicated that he/she used a mobile touch screen device weekly. Three subjects indicated that they had rarely or never used a touch screen device before. Once again, this is not representative of the general population. However, it should be noted that an continual authentication system based on touch dynamics will primarily be used by people who own such a device. In this case, it is preferable that the dataset only contain users who are used to a touch screen device so that we obtain realistic results.

5. **Effect of user specific characteristics** - Gender, dominant hand, mental state may have an effect on the user behavior. While the gender and dominant hand characteristics were collected as part of the dataset, we did not use it in the data analysis.

6. **Environmental variables -** While the environment was controlled, users were free to come in at any time of the data to complete their sessions. The time of day may have had an effect on the touch data collected.

4.12.3 Extending the results to other usage scenarios

Our app simulates a particular usage scenario where the user searches for a particular image. No literature exists that has studied usage context and its effect on the touch profile. Thus, it is possible that our results do not extend to other apps and activities such as web browsing, e-mail, watching video, etc. Future research will need to determine whether this is the case. In the event that it is not, usage context may prove to be a valuable dimension to leverage for more reliable and accurate authentication system.

4.12.4 Influence of sample size

We performed the data collection over a period of 2 weeks per subject. We finally used 31 subjects in this this experimental analysis. This sample size is sufficient for the statistical tests that we are conducting. However, a real-world setting has a significantly large number of users and impostors. Currently, further data collection is underway to further enlarge the dataset. One of the future areas of research on this expanded dataset is to analyze the impact on performance as the sample size increases.

4.12.5 Data density per test subject

While the data was collected from multiple test subjects, some subjects used far more strokes to complete a game. In order to elicit the most uniform results, we equalized the number of strokes used per subject to build a model. To do so, each subject's dataset was pruned such that the number of strokes per subject was the same as that of the subject with the lowest number of strokes. While this gives us the benefit of a uniform classifier, it must be noted that additional strokes are available for most users and were not used in the data analysis.

4.12.6 Normalized features

During data analysis, we noted that the pressure sensing and reading was different across the various devices. On some devices, the default pressure reading was non-zero. The upper and lower limit of the pressure value would vary. This variation may have played a role in helping distinguish between devices. One of the future areas of research is to study the effect of normalizing the pressure readings such that all devices show pressure readings within the same range.

4.13 Chapter Summary

This chapter investigated the underlying dynamics of touch based authentication. Specifically, we analyzed the effect of device size, user posture, and device manufacturer on a touch-based authentication system's performance using a locally collected dataset. Various parameters such as user environment, devices used, session length, etc. were controlled in order to remove confounding factors.

- Based on a benchmark analysis of five classification algorithms, we determined that the Random Forest algorithm was the most stable and reliable algorithm for our system.
- Our approach resulted in a classifier performance that is either better or equivalent to the benchmark performances reported in literature.
- This benchmark performance has been achieved by using fewer features than other studies in literature. For example, Frank et al. [24] used 27 features, Feng et al. [23] used 53 features and Serwadda et al. [67] used 28 features.
- We demonstrated that the user posture, device size, and device manufacturer have a significant effect on the classifier performance, with the device size having the greatest effect and the device manufacturer the least. We concluded that these factors must be accounted for when building a touch-based authentication system.
- These findings exposes the limitations of current state-of-the-art techniques for feature generation in touch dynamics. These attributes are ineffective when porting a user model from one device to another, from one posture to another, or from one device manufacturer to another device manufacturer. We provide suggestions for techniques to develop more robust attributes in Section 5.3.

Chapter 5
Conclusion

5.1 Comparing keystroke-dynamics and touch-dynamics

Keystroke dynamics has traditionally been associated with desktop and laptop computers. This is because a mechanical keyboard is a standard input device for computers and is vital to leverage this biometric modality. In a continual authentication scenario, a keystroke dynamics authentication system generates a user model based on the typing rhythm of digraphs, tri-graphs and/or n-graphs. When a string of characters is typed by the user, the system parses its constituent n-graphs and compares them to those present in the user model. A modern extended keyboard has 110-112 keys. A user model based on every possible n-graph is impractical as it will lead to an unreasonably large search space for the classifier. To mitigate this issue, only the most commonly typed n-graphs are incorporated into the user model, thus considerably reducing the search space.

Touch-based authentication, on the other hand, is tailored towards touch-screen devices as it relies solely upon the gestures executed by the user. As the touchscreen is the solitary medium of input and due to its relatively small size, the search space for a touch-based continual authentication system is relatively small compared to a keystroke-dynamics based system. This results in a simpler user model at the expense of lower authentication performance.

Both keystroke dynamics and touch dynamics are behavioral biometrics. Due to this, they are more susceptible to noise compared to physiological biometrics such as iris and fingerprint. For example, keystroke-dynamics based systems are affected by the user's mood, user posture, time keyboard type, computer specifications, and the software layers between the user and the system. Touch-based systems are similarly affected by the user's movement, mood, posture, and the device to device variations in touch sampling rates. However, modern touchscreen devices have a number of sensors such as an accelerometer, gyroscope and a GPS. Research has shown that augmenting touch-based authentication with other modalities provides improved performance [10]. Unlike desktop computers, the user's posture, movement and device location can be detected via these sensors. This data enables the development of user models with varying levels of granularity. Research investigating the effect of each of these modalities is currently lacking and merits further investigation.

Keystroke dynamics based authentication for cellphones has been explored since before the touchscreen era [17, 14, 51]. With the arrival of touchscreen devices, initial research was performed in incorporating keystroke dynamics when using PINs [38]. However, limited progress has been made in implementing continual authentication on a full-fledged software keyboard [63, 79, 29]. There are various reasons for this:

1. Unlike mechanical keyboards that have keyboard layout standards [3, 39, 40], software keyboards have not yet been standardized. A standardized layout allows for easier portability of user models across multiple devices. Currently, software keyboards vary across operating systems and device manufacturers making it difficult to develop a system that works across all keyboard layouts. Furthermore, even within a particular device and operating system environment, various developers offer their own keyboard layouts [59, 50] further complicating the user model development.

2. Due to the small screen size and the limited space to display the keyboard, developers use different strategies to incorporate character keys into the software keyboard. These strategies differ according to the software developer. Some developers use a special softkey that must be held down to pop-up a secondary keyboard to type special characters [50]. On the default Android keyboard, special characters are entered by holding down regular character keys for an extended period of time.

3. Modern software keyboard offer unique ways to type words. Apps such as Swiftkey [50] and Swype [59] allow the user to swipe through a string of keys and the app intelligently detects the word that the user wished to type. Using data from such gestures and character input techniques requires novel techniques for feature generation. The feature generation techniques from traditional keystroke dynamics cannot be ported to modern software keyboards. Recently, Trojahn and Ortmeier have proposed techniques for feature generation for such keyboards[75].

4. The Android operating system does not allow two programs to access the same touch data. Once a touch event is recorded by one program, it is 'destroyed' and cannot be read by another program. Due to this issue, it is not possible to develop a background keystroke-dynamics/touch-dynamics based authentication program using an unmodified Android operating system. Thus, all experiments conducted to determine the feasibility of touch based authentication can only be performed using simulated apps and environments. This severely limits the ability of practitioners to develop robust touch-based authentication systems.

5.2 Conclusion

Traditional password based access control systems are based on a 'what-you-know' paradigm. Keystroke dynamics, however, has an additional security layer based on the 'who-you-are' paradigm. This makes it more secure to impostor attacks. It seems unlikely that passwords will be replaced by another access control system in the near future. This provides a strong motivation to incorporate keystroke dynamics based biometrics into conventional systems as it does not require additional infrastructure for implementation. Our work provides an empirical analysis of the effect of user habituation on the user's typing behavior

and leveraging event sequences to improve authentication performance. These contributions lead to better keystroke dynamics based authentication systems.

Touch screen devices have experienced a rapid rise in popularity in recent years. Currently used security measures on these devices such as PINs and passwords have been imported from desktop computers. Comparatively, touch-based authentication is the most natural means to continually authenticate users on touch screen devices. Our work provides a rigorous empirical analysis of the effects of three important factors that affects the performance of a touch-based authentication system: user posture, device size, and device manufacturer. These contributions allow practitioners to recognize the limitations of the current state-of-the-art attributes and develop novel attributes that lead to more robust touch-based authentication systems.

To reiterate, this work provides three main contributions:

1. We analysed of the effect of user habituation in keystroke dynamics on a user's keystroke dynamics profile, its effect on inter-user profile separability with time, and its impact on authentication performance. We showed that user habituation has a statistically significant impact upon the user profile and the performance of keystroke dynamics based authentication schemes. Based on our findings, we tested several training models and concluded that the best model to improve the accuracy of keystroke dynamics authentication systems is one that retrains on the most recent set of keystroke entries.

2. We demonstrated that event sequences are an effective attribute for use in keystroke dynamics based authentication systems. The event sequence is the temporal sequence of all key-press and key-release events performed to type a string. This includes the key-press and key-release events of character keys and of special keys (Caps Lock, Left Shift, Right Shift, etc) that are used to modify the character key output. In contrast to the traditional approach in literature of ignoring these variations, we showed using empirical analysis that including variations in event sequences in keystroke dynamics based authentication systems leads to better performance and reliability. We also demonstrated that event sequences possess discriminatory information that is independent of typing proficiency. This is in contrast to currently existing keystroke dynamics based attributes, all of which rely on inter-user variations in typing proficiency.

3. We demonstrated that the user's posture, the device size and the device manufacturer have a significant impact on the authentication performance of a touch-based authentication system. We showed that the attributes used in current state-of-the-art touch-based authentication systems lead to a user model that is incapable of providing constant, reliable performance when any of the above-mentioned three factors are changed.

5.3 Future work

5.3.1 Keystroke dynamics

1. The KDS-2 dataset comes from a continual collection process and, in a future iteration, users will be asked to submit the same credentials to determine a trend between the commonality of a password and its effectiveness. Furthermore, we plan to ask users to observe each others typing rhythm and attempt to mimic it when entering impostor credentials.

2. With respect to event sequences, it is important to note that further research is needed to offer a better understanding of the user habituation behavior demonstrated in Section 3.9.6. While we have shown that event sequences differentiate the keystroke sequences of paired users, it is necessary to conduct the experiment over a larger user sample to obtain generalizable results.

3. In this work, we demonstrated the discriminatory power of event sequences and its independence from a user's typing speed. A promising area of research is the development of a novel keystroke dynamics authentication system that fuses traditional keystroke dynamics attributes with event sequences. Such a system would be able to fuse the discriminatory power of traditional attributes (that can distinguish based on typing speed) and event sequences (that are independent of typing speed) to deliver better authentication performance.

4. Our experimental results show that the password acclimatization period takes more than 30 entries after which the user uses fewer variations of event sequences to type the password. We theorize that some users probably exhibit a shorter acclimatization period compared to others. Furthermore, the password acclimatization period may also depend on the complexity of the string and the frequency with which users type it. A deeper investigation of this topic would be appropriate.

5.3.2 Touch dynamics

Touch-based authentication is a relatively new field with considerable scope for further research. The dataset created during the course of this work will be made publicly available in the near future. Within and without this dataset, there are a number of research areas that need to be investigated. Some of these are described below:

1. User habituation: The experiment was designed so that the effect of user habituation could be captured in the dataset. The test subjects were only allowed to take part in one session in a sitting. Due to this, the dataset incorporates touch data over many sittings. Initially, the users were unfamiliar with the app and would have exhibited a different behavior. However, as they familiarized to the app and the devices, we believe

that their gestures would have changed to a certain extent. An evaluation of this change in user behavior would provide insight into the amount of time it may take a user to settle into a steady pattern. This stable profile may be more robust and provide better authentication performance.

2. Effect of User Interface design on performance: As Section 4.8 showed, there is a significant drop in performance when a user model is ported to a larger or smaller device. We believe one of the major reasons for this is that the UI expands and contracts with screen size. This in turn effects the coordinates of the touch (such as start and stop points). Since this data forms one of the key features of the user model, it is natural for user models to be incompatible between devices of significantly different screen sizes. The effect of this problem could be reduced in one of two ways:

   (a) Transformation size-specific features by scaling them to the ported device. For example, if a user model from a larger tablet is ported to a smaller smartphone, the coordinates of a gestures start and stop points may be non-existent on a smartphone due to its smaller size. These coordinates could be scaled according to the smartphone's dimensions to approximate the user's behavior on a smaller device.

   (b) Instead of referring to a gesture's location by its absolute coordinates, referring to its location with reference to UI elements may be a more natural means to determine user behavior. We believe that the location where the users tend to execute gestures is dependent not on the physical screen device, but rather on the UI. For example, during our data collection, we observed that users prefer to touch the photos themselves and not on their side, even when they changed the screen orientation (and thus the screen aspect ratio) in different postures.

3. An analysis of the dataset revealed that each device employed its own scheme for representing pressure. Some devices used a scale of 0.00 to 1.00, while others functioned within a much narrower scale. While this functions as identifying characteristic for different devices, it also causes issues when a user model needs to be ported. One area of future research would be to normalize pressure readings according to the destination device. This would increase compatibility between different devices.

4. While each test subject took part in the same number of sessions, the number of strokes submitted by each user varied. This is because some users chose to use shorter strokes when navigating, thus employing more strokes to finish their session. The effect of the number of strokes was removed from this study in order to remove a confounding factor. However, the typical length of a user's stroke and the number of strokes used would form a useful feature that could be incorporated in a user's touch model. This topic requires further investigation.

5. While our dataset consists of both horizontal and vertical strokes stored separately, our classifier was trained on a combination of both. Other researchers have taken the approach of constructing separate classifiers for each stroke type [67]. This strategy may lead to better performance as it simplifies the data supplied to each classifier.

6. As described in Section 4.5, sequential groups of strokes were used in generating a training set. In our experiments, each group contained 5 strokes. It is possible that increasing the group size in the training set will lead to more reliable performance. Similarly, the increasing the group size in the testing set may lead to better results at the expense of greater time to authentication. The trade-off between these variables is an interesting area for further research.

Chapter 6
Appendix

A Benchmark analysis of classification algorithms

In Section 4.7, in order to determine the best performing classifier for our touch-based authentication system, we performed a benchmark analysis using our dataset using the following steps:

1. 50% of the user data was assigned to the testing set. The training and testing data was extracted as described in Section 4.5.
2. 10 training sets of varying sizes were created by varying the amount of training data from 10%-100% of the complete training set.
3. Each of the 10 training sets was used to create a user model based on the Random Forest classifier.
4. For every user, the 10 user models were tested against the testing set (which is of a constant size) and their EERs calculated.
5. The mean EER for all users at every training set size was calculated.

We tested five classification algorithms: Logistic Regression, SVM, Random Forest, Naive Bayes, and Multi-layer Perceptron. This section of the appendix lists the EERs calculated at Step 4 for all 31 users for each classification algorithm.

Table 35: EERs for Random Forest - Users 1-10

	Users									
TrainSet%	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
10	0.1743	0.133	0.2271	0.1147	0.172	0.1537	0.1238	0.0206	0.172	0.2179
20	0.0275	0.1009	0.117	0.1422	0.1629	0.1147	0.1078	0.0069	0.1238	0.0665
30	0.0527	0.0596	0.0963	0.0849	0.1124	0.094	0.1078	0.0069	0.1009	0.0757
40	0.0482	0.078	0.0757	0.078	0.1055	0.0872	0.078	0.0069	0.0665	0.0665
50	0.0367	0.055	0.0665	0.0596	0.0963	0.0757	0.0826	0.0023	0.0504	0.0459
60	0.0344	0.039	0.0642	0.0688	0.0734	0.0849	0.0619	0.0023	0.0459	0.0459
70	0.0413	0.0436	0.0642	0.0734	0.0665	0.0665	0.0573	0.0069	0.0344	0.0573
80	0.0344	0.0252	0.055	0.055	0.0573	0.0803	0.0619	0.0023	0.0298	0.039
90	0.0344	0.0321	0.0688	0.0665	0.055	0.0757	0.039	0.0023	0.039	0.0481
100	0.0275	0.0275	0.0482	0.0527	0.0413	0.078	0.039	0.0023	0.0367	0.0413

Table 36: EERs for Random Forest - Users 11-20

	Users									
TrainSet%	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**
10	0.3463	0.0757	0.0963	0.2225	0.0367	0.1881	0.1697	0.1927	0.2707	0.2179
20	0.1674	0.0849	0.0711	0.1858	0.0115	0.0986	0.1284	0.0803	0.1422	0.1216
30	0.1376	0.0206	0.0573	0.1606	0.0206	0.1124	0.1376	0.0872	0.1353	0.0895
40	0.1216	0.0482	0.0665	0.1376	0.016	0.0734	0.1078	0.0917	0.1193	0.0688
50	0.1147	0.0298	0.0482	0.1261	0.0069	0.0734	0.0986	0.0734	0.1124	0.0527
60	0.0757	0.0367	0.0482	0.1147	0.0069	0.0573	0.0757	0.0711	0.0826	0.0619
70	0.0895	0.0252	0.0459	0.1009	0.0115	0.0757	0.0894	0.0573	0.0917	0.0573
80	0.0894	0.0252	0.0573	0.0963	0.0115	0.0596	0.0917	0.0482	0.0573	0.0527
90	0.0803	0.016	0.0482	0.0894	0.0069	0.0596	0.078	0.0459	0.0596	0.0344
100	0.0573	0.016	0.0482	0.0826	0.0069	0.055	0.0734	0.0528	0.0665	0.0482

Table 37: EERs for Random Forest - Users 21-31

	Users										
TrainSet%	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**	**31**
10	0.172	0.2339	0.2064	0.1353	0.0825	0.2638	0.1583	0.2385	0.2225	0.2385	0.172
20	0.094	0.1904	0.1651	0.1101	0.0642	0.2133	0.1193	0.2385	0.1606	0.1697	0.1445
30	0.0665	0.0986	0.117	0.094	0.016	0.1399	0.0481	0.1721	0.1514	0.1606	0.0826
40	0.0803	0.1055	0.094	0.0986	0.0344	0.1169	0.0642	0.1743	0.1583	0.1307	0.1078
50	0.0527	0.0619	0.0711	0.0665	0.016	0.1147	0.0482	0.1399	0.1216	0.133	0.0872
60	0.0573	0.0665	0.0849	0.0642	0.0275	0.1009	0.0367	0.1124	0.1032	0.1078	0.1009
70	0.0596	0.0734	0.0711	0.0688	0.0298	0.0986	0.0482	0.1055	0.0895	0.0963	0.0803
80	0.0573	0.0619	0.0711	0.0642	0.0138	0.0963	0.039	0.1055	0.1124	0.1124	0.0803
90	0.0482	0.0757	0.0688	0.0436	0.0275	0.0963	0.0459	0.0849	0.0872	0.1009	0.0826
100	0.0482	0.078	0.0688	0.0436	0.0229	0.0826	0.0413	0.0963	0.094	0.1124	0.0711

Table 38: EERs for SVM - Users 1-10

	Users									
TrainSet%	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
10	0.3188	0.1559	0.4037	0.2271	0.211	0.1284	0.2408	0.273	0.1445	0.1949
20	0.0642	0.1307	0.1353	0.1904	0.1628	0.1812	0.1422	0.0114	0.1972	0.1307
30	0.1009	0.0573	0.1583	0.2982	0.1307	0.1353	0.1766	0.2637	0.1032	0.1927
40	0.0573	0.1009	0.1881	0.0872	0.1606	0.094	0.094	0.0137	0.117	0.1147
50	0.0482	0.0826	0.1376	0.1078	0.1904	0.1307	0.1262	0.0229	0.094	0.0895
60	0.0642	0.0734	0.1468	0.1216	0.1147	0.0642	0.1032	0.0069	0.0642	0.172
70	0.0711	0.1078	0.1445	0.1216	0.1399	0.0986	0.0917	0.0069	0.0803	0.1078
80	0.0757	0.0734	0.1055	0.0826	0.1147	0.117	0.1193	0.0046	0.0665	0.0849
90	0.0413	0.078	0.1101	0.0734	0.0779	0.094	0.0963	0.0138	0.0665	0.0619
100	0.0551	0.0482	0.0986	0.1124	0.1193	0.0849	0.1032	0.0137	0.0573	0.0734

Table 39: EERs for SVM - Users 11-20

	Users									
TrainSet%	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**
10	0.4404	0.1514	0.1445	0.1995	0.2729	0.2592	0.2202	0.3509	0.2523	0.227
20	0.3372	0.0849	0.1192	0.234	0.0206	0.2064	0.1674	0.211	0.211	0.1766
30	0.195	0.0367	0.0826	0.1881	0.1078	0.211	0.1674	0.1789	0.1904	0.1192
40	0.2133	0.0436	0.078	0.1491	0.0321	0.0986	0.1399	0.0963	0.1468	0.1537
50	0.1422	0.0206	0.0734	0.1835	0.0206	0.094	0.1537	0.1514	0.1537	0.1055
60	0.1399	0.0596	0.0413	0.1491	0.0115	0.1147	0.1055	0.117	0.1032	0.1284
70	0.1697	0.0871	0.0527	0.1537	0.0115	0.0803	0.0917	0.0963	0.1284	0.094
80	0.1124	0.094	0.0436	0.1468	0.0023	0.0849	0.0963	0.0895	0.1147	0.0826
90	0.1216	0.0436	0.039	0.1238	0.0183	0.0688	0.0757	0.094	0.1147	0.0688
100	0.094	0.0436	0.0344	0.1238	0.0092	0.0872	0.0917	0.0849	0.1009	0.0596

Table 40: EERs for SVM - Users 21-31

	Users										
TrainSet%	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**	**31**
10	0.1835	0.2156	0.3395	0.2156	0.1192	0.2431	0.1697	0.3165	0.2385	0.289	0.2683
20	0.1238	0.2202	0.1927	0.1422	0.1629	0.2523	0.1261	0.3303	0.1972	0.2959	0.1582
30	0.1124	0.1514	0.1674	0.1514	0.0596	0.2775	0.1192	0.3188	0.1605	0.2156	0.1376
40	0.1445	0.1651	0.1193	0.1606	0.0229	0.1903	0.0803	0.2362	0.1697	0.2294	0.1216
50	0.1124	0.1032	0.1651	0.0986	0.0482	0.1812	0.0665	0.2569	0.2156	0.1858	0.1147
60	0.094	0.0757	0.0917	0.0986	0.0367	0.2179	0.094	0.3096	0.133	0.1537	0.1399
70	0.117	0.0894	0.0734	0.1055	0.0436	0.1904	0.0619	0.2408	0.1307	0.1697	0.1284
80	0.0917	0.1124	0.1537	0.0688	0.0413	0.1628	0.0803	0.2339	0.1697	0.1216	0.0986
90	0.0803	0.1239	0.0963	0.0803	0.0344	0.1583	0.0665	0.211	0.1216	0.1399	0.0826
100	0.0734	0.0986	0.0849	0.078	0.0504	0.1261	0.0596	0.2041	0.1353	0.1674	0.0986

Table 41: EERs for Multi-layer Perceptron - Users 1-10

	Users									
TrainSet%	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
10	0.7454	0.5848	0.4495	0.7294	0.4633	0	0.7844	0.3578	0.422	0.4426
20	0.7339	0.7431	0.7638	0.75	0.6514	0.5918	0.75	0.4771	0	0.4863
30	0.5321	0.2317	0.4243	0.7133	0.5505	0.75	0.75	0.7179	0.5298	0.3303
40	0.7317	0.7339	0.7431	0.7546	0.4174	0.4931	0.7546	0.7385	0.5504	0.5183
50	0.7339	0.5229	0.5023	0.6881	0.6353	0.6743	0.75	0.516	0.5184	0.6881
60	0.75	0.6835	0.672	0.75	0.7431	0.7408	0.75	0.7179	0.4633	0.6514
70	0.7661	0.75	0.4243	0.75	0.711	0.5183	0.75	0.6376	0.7362	0.4725
80	0.7431	0.7408	0.4794	0.3417	0.6812	0.4564	0.75	0.7248	0.6996	0.5138
90	0.4702	0.7615	0.6055	0.4725	0.5321	0.7225	0.7316	0.7386	0.5825	0.6239
100	0.5505	0.5527	0.6698	0.7248	0.5482	0.4633	0.75	0.727	0.7202	0.6353

Table 42: EERs for Multi-layer Perceptron - Users 11-20

	Users									
TrainSet%	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**
10	0.75	0.3899	0.4839	0.7477	0.7385	0.6537	0.7454	0.5895	0.5688	0.539
20	0.5596	0.1675	0.4473	0.7018	0.6675	0.6973	0.7041	0.4885	0.5115	0.7523
30	0.4174	0.3624	0.3647	0.539	0.5619	0.5918	0.6651	0.75	0.4335	0.5046
40	0.6605	0.4908	0.3555	0.75	0.6216	0.7477	0.5069	0.7133	0.6009	0.406
50	0.4381	0.4656	0.4243	0.5665	0.75	0.75	0.3922	0.75	0.7454	0.5803
60	0.5321	0.4518	0.5138	0.7546	0.5803	0.75	0.6376	0.7408	0.4633	0.5046
70	0.6697	0.1606	0.5711	0.75	0.6973	0.7041	0.8119	0.6973	0.75	0.3555
80	0.6904	0.3739	0.5344	0.7019	0.6261	0.7477	0.5092	0.7133	0.5849	0.6605
90	0.7408	0.5023	0.3005	0.711	0.656	0.75	0.672	0.7684	0.75	0.578
100	0.6537	0.4725	0.2821	0.7408	0.6285	0.4519	0.633	0.7661	0.75	0.3692

Table 43: EERs for Multi-layer Perceptron - Users 21-31

	Users										
TrainSet%	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**	**31**
10	0.6996	0.4335	0.5619	0.3555	0.25	0.578	0.4243	0.3715	0.6032	0.5757	0.4013
20	0.4174	0.5895	0.5527	0.75	0.3119	0.6009	0.594	0.3876	0.4381	0.6858	0.7707
30	0.4014	0.7546	0.6537	0.5207	0.4358	0.383	0.3532	0.5619	0.6468	0.6216	0.6927
40	0.6261	0.75	0.6468	0.7248	0.5528	0.6261	0.4358	0.5459	0.4863	0.6812	0.5596
50	0.5069	0.7408	0.3647	0.2959	0.4954	0.5895	0.5665	0.5252	0.6697	0.4725	0.5
60	0.5734	0.6032	0.5229	0.5161	0.3463	0.6996	0.328	0.6972	0.5413	0.6101	0.7569
70	0.3784	0.75	0.6032	0.5069	0.4702	0.6582	0.4816	0.6743	0.4908	0.7064	0.4473
80	0.3624	0.5046	0.5436	0.4036	0.5826	0.4748	0.6491	0.5138	0.5505	0.6216	0.6353
90	0.5527	0.6812	0.5367	0.7431	0.5183	0.6583	0.4633	0.6766	0.672	0.4312	0.5894
100	0.5023	0.7523	0.3922	0.7523	0.4771	0.6239	0.6789	0.7431	0.6743	0.578	0.578

Table 44: EERs for Logistic Regression - Users 1-10

	Users									
TrainSet%	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
10	0.1514	0.1652	0.2156	0.1812	0.1697	0.2179	0.1927	0.0321	0.2913	0.2592
20	0.0642	0.1422	0.1995	0.1835	0.1583	0.195	0.1124	0.0367	0.2156	0.2179
30	0.094	0.1284	0.1835	0.1789	0.1651	0.1766	0.1193	0.0298	0.3005	0.2087
40	0.0688	0.1514	0.2041	0.1973	0.1743	0.1697	0.1101	0.0413	0.25	0.1949
50	0.0803	0.1376	0.1651	0.1537	0.211	0.1491	0.1147	0.0367	0.3142	0.2225
60	0.0826	0.1307	0.2018	0.1743	0.1766	0.1675	0.1147	0.0229	0.2661	0.211
70	0.0734	0.1307	0.2202	0.1674	0.1743	0.156	0.1124	0.0183	0.2684	0.2202
80	0.0803	0.1101	0.1835	0.172	0.1743	0.1697	0.1009	0.0275	0.2592	0.2018
90	0.055	0.1238	0.2064	0.1583	0.1651	0.1651	0.1147	0.0275	0.2707	0.2202
100	0.055	0.1399	0.1995	0.1743	0.1812	0.1743	0.1078	0.0275	0.2684	0.2064

Table 45: EERs for Logistic Regression - Users 11-20

	Users									
TrainSet%	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**
10	0.3165	0.1101	0.1789	0.3142	0.0573	0.234	0.1995	0.3784	0.2775	0.3417
20	0.2844	0.0734	0.1399	0.2133	0.0596	0.156	0.1881	0.2293	0.273	0.2271
30	0.25	0.0436	0.1193	0.234	0.0596	0.1262	0.1651	0.25	0.2408	0.234
40	0.2546	0.039	0.133	0.2087	0.0573	0.1307	0.1583	0.2293	0.2431	0.1904
50	0.2477	0.0596	0.1216	0.1927	0.0482	0.1399	0.156	0.2431	0.2821	0.1995
60	0.2569	0.039	0.1284	0.2133	0.0436	0.1215	0.1606	0.211	0.2454	0.188
70	0.2523	0.0436	0.1284	0.211	0.039	0.1307	0.1674	0.2271	0.2385	0.1904
80	0.2454	0.0413	0.1216	0.1927	0.0482	0.1284	0.1743	0.2454	0.2523	0.2225
90	0.2982	0.039	0.1307	0.2019	0.0436	0.1261	0.172	0.2271	0.2523	0.2064
100	0.25	0.0482	0.1262	0.1995	0.0459	0.1445	0.1628	0.2179	0.2408	0.2179

Table 46: EERs for Logistic Regression - Users 21-31

	Users										
TrainSet%	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**	**31**
10	0.2798	0.2317	0.2431	0.2638	0.2523	0.3326	0.3509	0.3647	0.2133	0.3463	0.3051
20	0.2409	0.2615	0.2202	0.2385	0.2477	0.3716	0.2661	0.3601	0.172	0.2982	0.195
30	0.1881	0.2156	0.2179	0.2018	0.2569	0.3142	0.2684	0.3372	0.1927	0.3601	0.2294
40	0.2362	0.2156	0.2317	0.2248	0.234	0.2982	0.2661	0.3372	0.1835	0.3372	0.2087
50	0.1812	0.195	0.2133	0.2156	0.2294	0.2936	0.2409	0.3555	0.1743	0.3234	0.2018
60	0.1651	0.2019	0.2385	0.1881	0.1972	0.2982	0.2546	0.3417	0.1812	0.3257	0.2339
70	0.172	0.211	0.2317	0.1927	0.2018	0.2982	0.2752	0.3372	0.1766	0.344	0.2087
80	0.1812	0.1927	0.234	0.1927	0.2271	0.2982	0.2684	0.3303	0.1743	0.3532	0.2179
90	0.1514	0.2156	0.2385	0.1927	0.2294	0.2867	0.2752	0.3532	0.1674	0.3372	0.2133
100	0.1491	0.2087	0.2408	0.195	0.2294	0.2798	0.2706	0.3326	0.1743	0.3509	0.2225

Table 47: EERs for NaiveBayes - Users 1-10

	Users									
TrainSet%	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
10	0.1307	0.1812	0.25	0.1789	0.1996	0.1766	0.1445	0.328	0.3005	0.25
20	0.1124	0.1583	0.1858	0.1399	0.1743	0.1697	0.1216	0.2798	0.25	0.2386
30	0.1193	0.1812	0.2064	0.156	0.1514	0.1651	0.1422	0.039	0.2982	0.2408
40	0.3211	0.1193	0.3486	0.1789	0.3808	0.1583	0.1606	0.039	0.2821	0.2317
50	0.3257	0.1514	0.2087	0.1445	0.1628	0.1697	0.1445	0.039	0.2638	0.2156
60	0.3257	0.1491	0.4289	0.3395	0.195	0.156	0.3532	0.0298	0.2706	0.2294
70	0.3463	0.1261	0.484	0.3532	0.3899	0.3326	0.3646	0.039	0.2408	0.2271
80	0.3807	0.3303	0.539	0.3555	0.3784	0.3578	0.3922	0.0298	0.2752	0.2363
90	0.3647	0.3303	0.5459	0.3555	0.3647	0.3716	0.406	0.0321	0.2523	0.2385
100	0.3647	0.328	0.5367	0.3555	0.3601	0.3509	0.3991	0.0321	0.2569	0.2385

Table 48: EERs for Naive Bayes - Users 11-20

	Users									
TrainSet%	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**
10	0.3349	0.1193	0.1307	0.2959	0.0321	0.25	0.1743	0.4083	0.3486	0.1996
20	0.3326	0.0849	0.0849	0.2798	0.039	0.4105	0.3853	0.4312	0.4518	0.1491
30	0.4771	0.305	0.1032	0.4473	0.0344	0.3899	0.3647	0.4128	0.3119	0.3257
40	0.4587	0.3027	0.1147	0.4449	0.039	0.4587	0.4427	0.4793	0.6055	0.1376
50	0.4518	0.1009	0.1101	0.2867	0.039	0.2316	0.1353	0.1927	0.2523	0.133
60	0.3693	0.0757	0.1124	0.4082	0.039	0.2271	0.3463	0.3968	0.2798	0.1261
70	0.4404	0.0757	0.1009	0.4266	0.039	0.4289	0.4197	0.461	0.5574	0.1353
80	0.4748	0.0848	0.1009	0.2936	0.039	0.4106	0.3991	0.4472	0.5	0.1284
90	0.4564	0.3004	0.1078	0.4014	0.039	0.4243	0.4037	0.4564	0.5482	0.1422
100	0.4587	0.2913	0.1055	0.3005	0.039	0.422	0.3968	0.4518	0.5275	0.1238

Table 49: EERs for Naive Bayes - Users 21-31

	Users										
TrainSet%	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**	**31**
10	0.4977	0.4839	0.5184	0.4587	0.4564	0.5092	0.4105	0.5481	0.4174	0.5734	0.4404
20	0.5527	0.6101	0.5321	0.5183	0.4794	0.4679	0.4839	0.6537	0.5183	0.6376	0.5
30	0.3807	0.2248	0.3601	0.3601	0.4312	0.2798	0.4014	0.4174	0.3876	0.4404	0.1812
40	0.2454	0.5551	0.4816	0.4357	0.4495	0.4404	0.3807	0.5803	0.4931	0.6078	0.4702
50	0.2683	0.4152	0.3555	0.3348	0.4083	0.3097	0.3647	0.3762	0.3693	0.4106	0.3716
60	0.2615	0.2477	0.1881	0.1376	0.2775	0.3211	0.3693	0.344	0.2248	0.2936	0.1835
70	0.3693	0.2477	0.4473	0.4335	0.4564	0.4174	0.4082	0.4977	0.4312	0.4197	0.4128
80	0.2294	0.4908	0.4083	0.3899	0.4266	0.3876	0.3991	0.5206	0.4312	0.3876	0.4174
90	0.367	0.2546	0.4105	0.3853	0.4266	0.3876	0.3945	0.4931	0.4197	0.3945	0.3991
100	0.2294	0.2569	0.4037	0.3807	0.4151	0.3876	0.3968	0.4679	0.4013	0.3899	0.3876

B Effect of device size on authentication performance

In Section 4.8, we performed an empirical analysis to measure the effect of device size on the authentication performance of a touch-based authentication system. In this experiment we restricted the user posture in order to remove a confounding variable. In order to study the effect of device size, we performed a statistical comparison using the following steps:

1. From the touch data for a given user and posture, 3 pairs of training and testing sets were extracted using the approach described in Section 4.5. Each set contained the same number of strokes and was constructed using the touch data from one of three devices (T10: 10" tablet, T7: 7" tablet and S3: 4.7" smartphone). These three devices, described in Section 4.2 are all made by Samsung. We excluded the HTC Evo from this experiment in order to keep the device manufacturer same for all devices under test. The training sets for a user are labeled $Train^{T10}$, $Train^{T7}$, and $Train^{S3}$. The testings set are labeled $Test_{T10}$, $Test_{T7}$, and $Test_{S3}$.

2. Three user models were constructed using $Train^{T10}$, $Train^{T7}$, and $Train^{S3}$ with the Random Forest classifier.

3. Each model was then tested on each of the three testing sets and the Equal Error Rate was calculated in every instance. The EER when training on Device X and testing on Device Y for User i is labeled $EER\_i^X_Y$. This procedure was performed over all users.

4. To test the effect of device size on authentication performance of a user model based on $Train^X$, the EER distributions EER^X_{T10}, EER^X_{T7}, EER^X_{S3} are compared to each other using a one-sided Student's t-test with Holm-Bonferroni correction.

This section of the appendix lists the EERs calculated at Step 3 for all 31 users.

Table 50: EERs when the posture is Posture 1

	T10			**T7**			**S3**		
User	**T10**	**T7**	**S3**	**T10**	**T7**	**S3**	**T10**	**T7**	**S3**
1	0.0275	0.6951	0.666	0.7018	0.0488	0.4547	0.7523	0.2805	0.1466
2	0.0275	0.7642	0.5603	0.5964	0.0752	0.2694	0.5298	0.2439	0.1121
3	0.0482	0.6443	0.6595	0.5619	0.0813	0.2952	0.5895	0.248	0.0581
4	0.0527	0.6972	0.6897	0.7271	0.0203	0.3491	0.7591	0.3557	0.0453
5	0.0413	0.6402	0.7737	0.6858	0.0833	0.3707	0.6789	0.1463	0.0798
6	0.078	0.5122	0.4116	0.367	0.0427	0.3319	0.445	0.4004	0.0538
7	0.039	0.5387	0.6314	0.5918	0.0528	0.3297	0.4633	0.2744	0.1013
8	0.0023	0.3963	0.334	0.3991	0.1341	0.2974	0.1147	0.2967	0.0991
9	0.0367	0.2703	0.4548	0.4404	0.0468	0.3211	0.4679	0.311	0.0431
10	0.0413	0.4715	0.4677	0.7638	0.0386	0.278	0.5275	0.1748	0.0538
11	0.0573	0.3455	0.638	0.445	0.0447	0.278	0.6353	0.2459	0.0905
12	0.016	0.6565	0.5603	0.8624	0.0671	0.4505	0.4174	0.3171	0.1702
13	0.0482	0.748	0.6379	0.6399	0.1199	0.6573	0.6606	0.4269	0.0819
14	0.0826	0.4858	0.5905	0.5734	0.0996	0.2759	0.5298	0.2337	0.0755
15	0.0069	0.4024	0.7479	0.9496	0.0224	0.4138	0.7798	0.3272	0.1013
16	0.055	0.5223	0.5517	0.5963	0.1057	0.4246	0.6284	0.4736	0.0927
17	0.0734	0.5488	0.7155	0.3119	0.0712	0.4246	0.5459	0.2724	0.0927
18	0.0528	0.4736	0.5366	0.5115	0.1057	0.3836	0.4931	0.3394	0.1142
19	0.0665	0.5915	0.3728	0.3807	0.0691	0.2845	0.2523	0.2114	0.1099
20	0.0482	0.6321	0.653	0.6697	0.0976	0.3729	0.672	0.3313	0.1142
21	0.0482	0.5244	0.681	0.7087	0.0528	0.3556	0.7133	0.2927	0.0754
22	0.078	0.7358	0.7715	0.5986	0.0305	0.2177	0.6812	0.3679	0.0646
23	0.0688	0.6687	0.416	0.5826	0.0427	0.2802	0.4885	0.2155	0.0581
24	0.0436	0.3354	0.6013	0.3302	0.0731	0.2931	0.3096	0.2541	0.1487
25	0.0229	0.8008	0.7651	0.7569	0.0244	0.2586	0.6881	0.1931	0.0776
26	0.0826	0.3516	0.3534	0.4886	0.0508	0.2974	0.4909	0.4492	0.0776
27	0.0413	0.3679	0.4806	0.3716	0.067	0.4224	0.4587	0.4695	0.0647
28	0.0963	0.7601	0.6229	0.75	0.0732	0.3319	0.5206	0.2459	0.0754
29	0.094	0.5732	0.5862	0.5803	0.0691	0.2414	0.4886	0.2947	0.0647
30	0.1124	0.6423	0.6056	0.5413	0.0773	0.3944	0.617	0.2459	0.1121
31	0.0711	0.7439	0.6573	0.7271	0.0711	0.1746	0.656	0.2134	0.0776

Table 51: EERs when the posture is Posture 2

	T10			**T7**			**S3**		
User	**T10**	**T7**	**S3**	**T10**	**T7**	**S3**	**T10**	**T7**	**S3**
1	0.057	0.5659	0.5915	0.7807	0.025	0.4978	0.8048	0.3477	0.0938
2	0.0483	0.5955	0.558	0.761	0.0478	0.6875	0.6952	0.3068	0.0915
3	0.0746	0.7591	0.7054	0.6448	0.0613	0.404	0.5855	0.2477	0.0804
4	0.0636	0.425	0.5313	0.6733	0.0455	0.1473	0.6623	0.2227	0.0201
5	0.0439	0.7296	0.6228	0.7149	0.0478	0.192	0.739	0.225	0.0424
6	0.0592	0.3681	0.6205	0.4101	0.0455	0.3772	0.5373	0.3477	0.0647
7	0.0153	0.4296	0.4174	0.5921	0.041	0.1674	0.489	0.1432	0.0246
8	0.0373	0.5523	0.7009	0.3969	0.0977	0.4464	0.4342	0.2568	0.0603
9	0.0877	0.3159	0.5268	0.5285	0.041	0.2366	0.5943	0.275	0.0603
10	0.0241	0.7068	0.4955	0.4781	0.0636	0.2098	0.5767	0.2114	0.0402
11	0.0175	0.5818	0.4152	0.4233	0.0387	0.279	0.4145	0.3182	0.0558
12	0.0504	0.1955	0.3839	0.7741	0.0863	0.3884	0.4715	0.2341	0.1004
13	0.0658	0.35	0.6607	0.3377	0.0818	0.3103	0.261	0.2432	0.1094
14	0.0548	0.4295	0.587	0.1667	0.0273	0.4353	0.4693	0.1659	0.0982
15	0.0461	0.7727	0.7567	0.5768	0.0273	0.7433	0.4825	0.3068	0.029
16	0.0351	0.2318	0.4263	0.4277	0.1091	0.4754	0.6776	0.3228	0.1451
17	0.0483	0.4841	0.4196	0.7259	0.0205	0.1696	0.5636	0.2159	0.0647
18	0.0877	0.6568	0.5446	0.3399	0.0341	0.3616	0.4868	0.2364	0.0603
19	0.0285	0.3728	0.6808	0.3289	0.1023	0.3259	0.2938	0.3046	0.0893
20	0.0746	0.6886	0.5357	0.7412	0.0568	0.4152	0.7018	0.35	0.1138
21	0.0921	0.5886	0.5625	0.6338	0.0591	0.2322	0.6733	0.1273	0.0714
22	0.0197	0.2863	0.4241	0.4737	0.0455	0.3594	0.6075	0.3227	0.0513
23	0.0219	0.6909	0.7879	0.4671	0.0341	0.2299	0.5373	0.2409	0.0513
24	0.0746	0.5932	0.4688	0.3048	0.075	0.3281	0.3004	0.1886	0.1094
25	0.0373	0.4591	0.5179	0.6732	0.0068	0.1965	0.6338	0.1545	0.0157
26	0.0592	0.4341	0.6629	0.443	0.0591	0.5446	0.5263	0.4864	0.0558
27	0.0987	0.4319	0.6295	0.6294	0.075	0.3169	0.546	0.3795	0.0536
28	0.0899	0.5046	0.5826	0.4649	0.0682	0.3482	0.511	0.2795	0.1071
29	0.0285	0.7705	0.6161	0.4583	0.0841	0.259	0.4671	0.3182	0.0893
30	0.0285	0.7	0.5514	0.4605	0.0478	0.3103	0.5175	0.4296	0.0826
31	0.0285	0.7614	0.6071	0.3925	0.0478	0.2388	0.6185	0.3205	0.0826

Table 52: EERs when the posture is Posture 3

	T10			**T7**			**S3**		
User	**T10**	**T7**	**S3**	**T10**	**T7**	**S3**	**T10**	**T7**	**S3**
1	0.0302	0.7441	0.574	0.8922	0.0413	0.382	0.8729	0.3386	0.092
2	0.0323	0.4193	0.748	0.4978	0.0571	0.49	0.528	0.2854	0.086
3	0.0345	0.6319	0.568	0.569	0.0571	0.264	0.6466	0.189	0.042
4	0.0388	0.5926	0.548	0.6099	0.0256	0.152	0.6444	0.1398	0.016
5	0.0302	0.6929	0.592	0.653	0.0846	0.27	0.6724	0.1831	0.072
6	0.0539	0.4468	0.43	0.4547	0.0532	0.388	0.4332	0.3347	0.048
7	0.0366	0.7854	0.858	0.7327	0.0531	0.168	0.5496	0.1772	0.054
8	0.0108	0.4252	0.732	0.5345	0.0787	0.302	0.597	0.3307	0.056
9	0.0151	0.4528	0.324	0.5022	0.0787	0.324	0.306	0.3347	0.09
10	0.0215	0.5728	0.4	0.5798	0.0275	0.226	0.487	0.1359	0.064
11	0.0474	0.4862	0.364	0.556	0.0728	0.336	0.3578	0.3189	0.058
12	0.0108	0.7441	0.386	0.7586	0.1004	0.332	0.4979	0.3563	0.098
13	0.0151	0.6457	0.384	0.4526	0.063	0.39	0.4052	0.4232	0.084
14	0.0388	0.2795	0.344	0.2974	0.0571	0.228	0.2694	0.2401	0.074
15	0.0215	0.3347	0.702	0.2758	0.0177	0.668	0.3707	0.3268	0.034
16	0.041	0.4961	0.572	0.5043	0.0866	0.544	0.472	0.374	0.124
17	0.056	0.7815	0.698	0.5086	0.0177	0.228	0.5323	0.2027	0.02
18	0.0151	0.3819	0.38	0.2177	0.0492	0.364	0.3815	0.3701	0.1
19	0.0474	0.5807	0.506	0.5151	0.0689	0.284	0.3793	0.378	0.126
20	0.069	0.5334	0.592	0.6617	0.0965	0.216	0.6789	0.3583	0.062
21	0.0323	0.2618	0.46	0.3599	0.0216	0.122	0.3793	0.1221	0.05
22	0.0862	0.5748	0.52	0.5065	0.0079	0.386	0.4504	0.3248	0.068
23	0.0711	0.5728	0.476	0.5	0.0177	0.27	0.4892	0.2008	0.068
24	0.0194	0.2716	0.328	0.2543	0.061	0.272	0.2436	0.1949	0.05
25	0.0281	0.8228	0.854	0.8384	0.0197	0.164	0.7586	0.0886	0.03
26	0.069	0.5748	0.466	0.5754	0.0866	0.576	0.5281	0.3288	0.042
27	0.0151	0.7638	0.452	0.6034	0.0669	0.346	0.4397	0.5059	0.058
28	0.0194	0.5236	0.67	0.6271	0.0807	0.298	0.6358	0.3327	0.056
29	0.0538	0.5059	0.642	0.5668	0.0394	0.358	0.6229	0.3405	0.066
30	0.1013	0.6319	0.612	0.5582	0.0492	0.31	0.5517	0.4035	0.074
31	0.0172	0.8445	0.672	0.8965	0.0413	0.32	0.8836	0.3169	0.084

C Effect of posture on authentication performance

In Section 4.9, we performed an empirical analysis to measure the effect of user posture on the authentication performance of a touch-based authentication system. In this experiment we restricted the device size in order to remove a confounding variable. In order to study the effect of user posture, we performed a statistical comparison using the following steps:

1. From the touch data for a given user and device, 3 pairs of training and testing sets were extracted using the interleaved approach. The training and testing sets contained the same number of strokes.

2. The training/testing sets were extracted using the touch data from three postures (P1: device on table, P2: device held in portrait orientation and P3: device held in landscape orientation). The training sets for a user are labeled $Train^{P1}$, $Train^{P2}$, and $Train^{P3}$. The testings set are labeled $Test_{P1}$, $Test_{P2}$, and $Test_{P3}$.

3. Three user models were constructed using $Train^{P1}$, $Train^{P2}$, and $Train^{P3}$.

4. Each model was then tested on each of the three testing sets and the equal error rate was calculated in every instance. The EER when training on Posture X and testing on Posture Y for User i is labeled $EER\_i_Y^X$. When this procedure is performed over all users, it provides a distribution of EER scores as shown in Table 27.

5. To test the effect of posture on authentication performance of a user model based on Posture X, the EER distributions EER_{P1}^X, EER_{P2}^X, EER_{P3}^X are compared to each other using a one-sided Student's t-test with Holm-Bonferroni correction.

This section of the appendix lists the EERs calculated at Step 4 for all 31 users.

Table 53: EERs when the device type is Device 1 (T10)

	Posture 1			Posture 2			Posture 3		
User	**P1**	**P2**	**P3**	**P1**	**P2**	**P3**	**P1**	**P2**	**P3**
1	0.0275	0.1404	0.0926	0.0826	0.0285	0.0798	0.0917	0.1272	0.0302
2	0.0275	0.2149	0.4914	0.1445	0.0285	0.4547	0.4725	0.4321	0.0323
3	0.0482	0.1184	0.2435	0.1216	0.057	0.2909	0.4106	0.5066	0.0345
4	0.0527	0.1118	0.2414	0.117	0.0483	0.2414	0.1422	0.1206	0.0388
5	0.0413	0.2434	0.3233	0.2271	0.0746	0.2608	0.4037	0.4145	0.0302
6	0.078	0.1206	0.472	0.1193	0.0636	0.4138	0.3968	0.3772	0.0539
7	0.039	0.1535	0.1659	0.1858	0.0439	0.1271	0.2569	0.1929	0.0366
8	0.0023	0.6732	0.9483	0.2386	0.0592	0.1314	0.9473	0.1667	0.0108
9	0.0367	0.0789	0.2155	0.2087	0.0153	0.2091	0.1927	0.1118	0.0151
10	0.0413	0.1732	0.3233	0.3188	0.0373	0.3168	0.3692	0.4342	0.0215
11	0.0573	0.2193	0.5	0.1881	0.0877	0.513	0.4243	0.5395	0.0474
12	0.016	0.1272	0.1444	0.0573	0.0241	0.0646	0.133	0.1426	0.0108
13	0.0482	0.1535	0.1379	0.0619	0.0175	0.0927	0.2133	0.1601	0.0151
14	0.0826	0.2412	0.5345	0.2844	0.0504	0.4569	0.3326	0.3312	0.0388
15	0.0069	0.682	0.9138	0.3876	0.0658	0.0474	0.8371	0.1535	0.0215
16	0.055	0.0723	0.4526	0.0895	0.0548	0.1875	0.2477	0.2456	0.041
17	0.0734	0.0658	0.1573	0.1238	0.0461	0.1659	0.211	0.1404	0.056
18	0.0528	0.2938	0.6142	0.2225	0.0351	0.2134	0.5848	0.3838	0.0151
19	0.0665	0.114	0.2241	0.2064	0.0483	0.3038	0.4014	0.3312	0.0474
20	0.0482	0.2193	0.2974	0.1812	0.0877	0.3664	0.4885	0.4364	0.069
21	0.0482	0.1316	0.1401	0.1812	0.0285	0.1854	0.4106	0.329	0.0323
22	0.078	0.1886	0.3858	0.1491	0.0746	0.3922	0.6101	0.557	0.0862
23	0.0688	0.1798	0.4354	0.172	0.0921	0.4267	0.5665	0.4978	0.0711
24	0.0436	0.0811	0.459	0.2041	0.0197	0.1207	0.3624	0.2281	0.0194
25	0.0229	0.0373	0.2608	0.1032	0.0219	0.1638	0.4335	0.3882	0.0281
26	0.0826	0.3048	0.5539	0.2844	0.0746	0.4268	0.3876	0.4978	0.069
27	0.0413	0.3355	0.4203	0.594	0.0373	0.0711	0.7202	0.1097	0.0151
28	0.0963	0.3574	0.388	0.3761	0.0592	0.1659	0.4908	0.2215	0.0194
29	0.094	0.193	0.2263	0.1812	0.0987	0.1918	0.25	0.3092	0.0538
30	0.1124	0.3465	0.4699	0.3074	0.0899	0.2845	0.4312	0.2961	0.1013
31	0.0711	0.1733	0.1832	0.3234	0.0285	0.056	0.75	0.1601	0.0172

Table 54: EERs when the device type is Device 2 (T7)

	Posture 1			Posture 2			Posture 3		
User	**P1**	**P2**	**P3**	**P1**	**P2**	**P3**	**P1**	**P2**	**P3**
1	0.0488	0.075	0.4094	0.124	0.025	0.2697	0.374	0.4091	0.0413
2	0.0752	0.15	0.3071	0.2195	0.0478	0.1831	0.2256	0.1364	0.0571
3	0.0813	0.1455	0.2972	0.1972	0.0613	0.2146	0.2988	0.2046	0.0571
4	0.0203	0.159	0.2697	0.2947	0.0455	0.187	0.378	0.2591	0.0256
5	0.0833	0.1841	0.187	0.1931	0.0478	0.1693	0.2907	0.2727	0.0846
6	0.0427	0.1318	0.439	0.1423	0.0455	0.4803	0.4797	0.5386	0.0532
7	0.0528	0.6023	0.6555	0.4228	0.041	0.1988	0.4045	0.1387	0.0531
8	0.1341	0.2613	0.4015	0.2297	0.0977	0.2894	0.3699	0.2841	0.0787
9	0.0468	0.2682	0.498	0.1423	0.041	0.3504	0.3598	0.2591	0.0787
10	0.0386	0.2591	0.3169	0.1199	0.0636	0.2854	0.2724	0.3296	0.0275
11	0.0447	0.2455	0.4272	0.1707	0.0387	0.439	0.4227	0.3477	0.0728
12	0.0671	0.2068	0.3583	0.1646	0.0863	0.6161	0.5122	0.5159	0.1004
13	0.1199	0.1568	0.3543	0.3171	0.0818	0.3898	0.3476	0.2909	0.063
14	0.0996	0.2114	0.1988	0.2825	0.0273	0.2067	0.376	0.1636	0.0571
15	0.0224	0.1137	0.2303	0.122	0.0273	0.2244	0.372	0.275	0.0177
16	0.1057	0.3659	0.4409	0.3435	0.1091	0.4291	0.4248	0.3568	0.0866
17	0.0712	0.6409	0.7854	0.5305	0.0205	0.0335	0.6098	0.0795	0.0177
18	0.1057	0.3613	0.2303	0.4492	0.0341	0.4764	0.376	0.2137	0.0492
19	0.0691	0.1432	0.2697	0.2276	0.1023	0.2205	0.3252	0.3	0.0689
20	0.0976	0.2704	0.2421	0.248	0.0568	0.185	0.2845	0.2182	0.0965
21	0.0528	0.3523	0.4527	0.1728	0.0591	0.0788	0.7459	0.4841	0.0216
22	0.0305	0.1387	0.1929	0.1118	0.0455	0.187	0.1403	0.2205	0.0079
23	0.0427	0.1636	0.1831	0.065	0.0341	0.1496	0.1992	0.1773	0.0177
24	0.0731	0.175	0.2854	0.2033	0.075	0.3012	0.3333	0.2796	0.061
25	0.0244	0.0478	0.0335	0.0996	0.0068	0.0452	0.0976	0.0296	0.0197
26	0.0508	0.2227	0.622	0.1098	0.0591	0.4429	0.5915	0.4296	0.0866
27	0.067	0.1909	0.2834	0.1483	0.075	0.2323	0.2459	0.2136	0.0669
28	0.0732	0.2659	0.2579	0.1626	0.0682	0.191	0.1382	0.1818	0.0807
29	0.0691	0.4273	0.3937	0.2561	0.0841	0.2677	0.4533	0.3705	0.0394
30	0.0773	0.125	0.374	0.1301	0.0478	0.2973	0.3578	0.3887	0.0492
31	0.0711	0.1659	0.1417	0.2256	0.0478	0.1161	0.2744	0.1409	0.0413

Table 55: EERs when the device type is Device 3 (S3)

	Posture 1			**Posture 2**			**Posture 3**		
User	**P1**	**P2**	**P3**	**P1**	**P2**	**P3**	**P1**	**P2**	**P3**
1	0.1466	0.4063	0.454	0.2996	0.0938	0.426	0.3923	0.4442	0.092
2	0.1121	0.2098	0.288	0.2177	0.0915	0.534	0.3707	0.2857	0.086
3	0.0581	0.25	0.346	0.1207	0.0804	0.53	0.2802	0.4955	0.042
4	0.0453	0.0513	0.278	0.3362	0.0201	0.2	0.3685	0.2143	0.016
5	0.0798	0.1741	0.184	0.2414	0.0424	0.26	0.2263	0.1875	0.072
6	0.0538	0.3214	0.528	0.2521	0.0647	0.428	0.5237	0.4576	0.048
7	0.1013	0.1339	0.286	0.1659	0.0246	0.338	0.2328	0.2121	0.054
8	0.0991	0.1808	0.346	0.2845	0.0603	0.544	0.6875	0.5826	0.056
9	0.0431	0.0937	0.308	0.0884	0.0603	0.312	0.2802	0.3014	0.09
10	0.0538	0.1384	0.208	0.1314	0.0402	0.278	0.3469	0.3862	0.064
11	0.0905	0.1161	0.436	0.1293	0.0558	0.382	0.4418	0.3281	0.058
12	0.1702	0.2143	0.5	0.2241	0.1004	0.574	0.4763	0.5268	0.098
13	0.0819	0.3951	0.368	0.5172	0.1094	0.668	0.4892	0.5402	0.084
14	0.0755	0.2165	0.268	0.209	0.0982	0.522	0.334	0.3169	0.074
15	0.1013	0.192	0.554	0.2758	0.029	0.588	0.4634	0.5268	0.034
16	0.0927	0.4308	0.478	0.3513	0.1451	0.41	0.4957	0.3482	0.124
17	0.0927	0.2232	0.254	0.2823	0.0647	0.112	0.4354	0.2701	0.02
18	0.1142	0.1786	0.478	0.2608	0.0603	0.398	0.4289	0.3058	0.1
19	0.1099	0.1607	0.458	0.1789	0.0893	0.356	0.334	0.3058	0.126
20	0.1142	0.3214	0.59	0.2393	0.1138	0.36	0.3362	0.3482	0.062
21	0.0754	0.1429	0.244	0.3082	0.0714	0.428	0.6961	0.7299	0.05
22	0.0646	0.1786	0.426	0.1875	0.0513	0.47	0.5797	0.5736	0.068
23	0.0581	0.0647	0.234	0.1207	0.0513	0.24	0.278	0.2567	0.068
24	0.1487	0.221	0.22	0.2091	0.1094	0.388	0.4612	0.4888	0.05
25	0.0776	0.0424	0.084	0.1229	0.0157	0.082	0.1789	0.029	0.03
26	0.0776	0.1897	0.134	0.1746	0.0558	0.126	0.2371	0.1853	0.042
27	0.0647	0.1987	0.314	0.222	0.0536	0.37	0.4591	0.433	0.058
28	0.0754	0.1629	0.25	0.1358	0.1071	0.24	0.4267	0.4531	0.056
29	0.0647	0.1361	0.144	0.1078	0.0893	0.186	0.2306	0.2366	0.066
30	0.1121	0.2232	0.354	0.2241	0.0826	0.298	0.4439	0.3795	0.074
31	0.0776	0.1295	0.416	0.0905	0.0826	0.428	0.4224	0.3348	0.084

D Effect of manufacturer on authentication performance

In Section 4.10, we performed an empirical analysis to measure the effect of device manufacturer on the authentication performance of a touch-based authentication system. In this experiment, we controlled the device size and user posture in order to remove a confounding variable. In order to study the effect of device size, we performed a statistical comparison using the following steps:

1. From the touch data for a given user and posture, 2 pairs of training and testing sets were extracted using the interleaved approach. The training and testing sets contained the same number of strokes.

2. The training/testing sets were extracted using the touch data from two devices (S3 and EVO). The training sets for a user are labeled $Train^{S3}$, and $Train^{EVO}$. The testings set are labeled $Test_{S3}$, and $Test_{EVO}$.

3. Two user models were constructed using $Train^{S3}$, and $Train^{EVO}$.

4. Each model was then tested on each of the two testing sets and the equal error rate was calculated in every instance. The EER when training on Device X and testing on Device Y for User i is labeled $EER\_i_Y^X$. When this procedure is performed over all users, it provides a distribution of EER scores as shown in Table 30.

5. To test the effect of device size on authentication performance of a user model based on Device X, the EER distributions EER_{S3}^X and EER_{EVO}^X are compared to each other using Student's t-test with Holm-Bonferroni correction.

This section of the appendix lists the EERs calculated at Step 4 for all 31 users.

Table 56: EERs when the training device type is Device 3 (S3)

	Posture 1		**Posture 2**		**Posture 3**	
User	**S3**	**EVO**	**S3**	**EVO**	**S3**	**EVO**
1	0.1466	0.1802	0.0938	0.25	0.092	0.1859
2	0.1121	0.178	0.0915	0.1414	0.086	0.2179
3	0.0581	0.1036	0.0804	0.2323	0.042	0.1624
4	0.0453	0.0564	0.0201	0.0429	0.016	0.0385
5	0.0798	0.1306	0.0424	0.1515	0.072	0.2201
6	0.0538	0.25	0.0647	0.2399	0.048	0.2243
7	0.1013	0.1148	0.0246	0.1035	0.054	0.1645
8	0.0991	0.1599	0.0603	0.1111	0.056	0.1368
9	0.0431	0.1509	0.0603	0.0732	0.09	0.1325
10	0.0538	0.187	0.0402	0.1212	0.064	0.0577
11	0.0905	0.1645	0.0558	0.1389	0.058	0.1261
12	0.1702	0.2387	0.1004	0.3358	0.098	0.1966
13	0.0819	0.223	0.1094	0.202	0.084	0.156
14	0.0755	0.2139	0.0982	0.1161	0.074	0.1837
15	0.1013	0.1261	0.029	0.1187	0.034	0.0834
16	0.0927	0.3559	0.1451	0.2626	0.124	0.235
17	0.0927	0.1261	0.0647	0.0732	0.02	0.0834
18	0.1142	0.1735	0.0603	0.2045	0.1	0.1752
19	0.1099	0.223	0.0893	0.2449	0.126	0.2072
20	0.1142	0.1892	0.1138	0.2045	0.062	0.1602
21	0.0754	0.169	0.0714	0.0808	0.05	0.1026
22	0.0646	0.1261	0.0513	0.0757	0.068	0.0727
23	0.0581	0.0788	0.0513	0.1237	0.068	0.1304
24	0.1487	0.2319	0.1094	0.1742	0.05	0.1453
25	0.0776	0.1216	0.0157	0.0328	0.03	0.0427
26	0.0776	0.1148	0.0558	0.0808	0.042	0.0513
27	0.0647	0.178	0.0536	0.1086	0.058	0.1132
28	0.0754	0.1238	0.1071	0.106	0.056	0.1389
29	0.0647	0.1373	0.0893	0.1161	0.066	0.1111
30	0.1121	0.1982	0.0826	0.1464	0.074	0.1218
31	0.0776	0.0924	0.0826	0.1035	0.084	0.1517

Table 57: EERs when the training device type is Device 4 (Evo)

	Posture 1		**Posture 2**		**Posture 3**	
User	**S3**	**EVO**	**S3**	**EVO**	**S3**	**EVO**
1	0.2091	0.1081	0.3817	0.106	0.476	0.0748
2	0.2004	0.1193	0.2545	0.101	0.216	0.1026
3	0.1509	0.036	0.1942	0.0581	0.162	0.0791
4	0.2393	0.0292	0.0402	0.0025	0.056	0.0149
5	0.1509	0.0721	0.2299	0.1111	0.222	0.0705
6	0.153	0.0699	0.2009	0.0606	0.198	0.0556
7	0.1185	0.0315	0.1518	0.0555	0.122	0.0491
8	0.1573	0.1103	0.1786	0.0656	0.2	0.032
9	0.0905	0.0968	0.0982	0.0328	0.152	0.0663
10	0.1617	0.0879	0.1563	0.0581	0.09	0.032
11	0.153	0.0586	0.1317	0.0707	0.152	0.0427
12	0.2737	0.1081	0.2746	0.1288	0.358	0.1133
13	0.1358	0.1351	0.1964	0.0606	0.282	0.0812
14	0.1789	0.1103	0.1429	0.0454	0.188	0.0106
15	0.1789	0.0338	0.1339	0.053	0.102	0.0128
16	0.3879	0.16	0.279	0.1161	0.22	0.1261
17	0.2198	0.0541	0.1317	0.0151	0.06	0.0085
18	0.181	0.1126	0.1607	0.0959	0.13	0.0491
19	0.1703	0.1103	0.2567	0.0505	0.244	0.094
20	0.1961	0.1081	0.2522	0.0732	0.162	0.062
21	0.1745	0.0699	0.1629	0.053	0.07	0.0406
22	0.1509	0.0744	0.1004	0.0328	0.064	0.0149
23	0.1616	0.0833	0.1027	0.0656	0.114	0.0299
24	0.278	0.1283	0.2567	0.106	0.148	0.0855
25	0.0884	0.0473	0.0379	0.0227	0.046	0.0192
26	0.1314	0.027	0.1428	0.0328	0.124	0.0085
27	0.1422	0.0564	0.1496	0.0581	0.182	0.0577
28	0.1401	0.0788	0.183	0.0833	0.186	0.0705
29	0.1099	0.0721	0.1183	0.053	0.108	0.0577
30	0.2564	0.0923	0.1741	0.1086	0.178	0.0555
31	0.0969	0.0428	0.1161	0.0858	0.15	0.1004

E Publications

E.1 Journal papers and articles

1. Syed, Z.; Cukic, B., "Continual Authentication," *Biometric Technology Today*, June 2014

2. Syed, Z.; Banerjee, S.; Cukic, B., "Pointer-Based recognition," *Encyclopedia of Biometrics*, 2014

3. Banerjee, S.; Syed, Z.; Cukic, B., "Keystroke recognition," *Encyclopedia of Biometrics*, 2014

4. Syed, Z.; Cukic, B., "Normalizing Feature Vector Structure in Keystroke-dynamics Authentication Systems," *Software Quality Journal*, 2014 [Submitted]

5. Syed, Z.; Banerjee, S.; Cukic, B., "Statistical Evaluation of User Habituation in Keystroke Dynamics," *IEEE Trans. on Dependable and Secure Computing (TDSC)*, 2014 [Submitted]

E.2 Conference publications

1. Banerjee, S.; Helmick, J.; Syed, Z.; Cukic, B., "A scalable approach to the triage of primary and duplicate problem reports", *ACM SIGSOFT Int'l Symp. on the Foundations of Software Engineering (FSE)* [Submitted]

2. Syed, Z.; Banerjee, S.; Cukic, B., "Leveraging variations in event sequences in keystroke-dynamics authentication systems," *IEEE Int'l Symp. on High-Assurance Systems Engineering (HASE)*, 9-11 Jan. 2014

3. Banerjee, S.; Syed, Z.; Helmick, J.; Cukic, B., "A Fusion Approach For Classifying Duplicate Problem Reports," *IEEE Int'l Symp. on Software Reliability Engineering (ISSRE)*, 4-7 Nov. 2013

4. Syed, Z.; Banerjee, S.; Qi Cheng; Cukic, B., "Effects of User Habituation in Keystroke Dynamics on Password Security Policy," *IEEE Int'l Symp. on High-Assurance Systems Engineering (HASE)*, vol., no., pp.352,359, 10-12 Nov. 2011 (Best Paper Award)

5. Syed, Z.A. and Noore, A., "Performance Optimization to Alleviate I/O Constraints in Designing Large FPGA Shifters," *IEICE Electronics Express*, vol. 5, no. 1, pp. 29-34, 2008

6. Syed, Z. and Cukic, B., "Effects of User Posture and Device Variation on Touch-based Authentication on Portable Devices," [In Preparation]

Bibliography

References

[1] Ahmed Awad E Ahmed and Issa Traore. A new biometric technology based on mouse dynamics. *Dependable and Secure Computing, IEEE Transactions on*, 4(3):165–179, 2007. 9, 10, 11, 12

[2] Jeffrey D Allen. *An analysis of pressure-based keystroke dynamics algorithms*. PhD thesis, Southern Methodist University, 2010. 65

[3] ANSI-INCITS-154-1988. Office machines and supplies - alphanumeric machines - keyboard arrangement, 1988. 53, 96

[4] Lívia CF Araújo, Luiz HR Sucupira Jr, Miguel Gustavo Lizarraga, Lee Luan Ling, and João Baptista T Yabu-Uti. User authentication through typing biometrics features. *Signal Processing, IEEE Transactions on*, 53(2):851–855, 2005. 5, 6

[5] Salil P Banerjee and Damon L Woodard. Biometric authentication and identification using keystroke dynamics: A survey. *Journal of Pattern Recognition Research*, 7(1):116–139, 2012. 6, 64

[6] Nick Bartlow and Bojan Cukic. Evaluating the reliability of credential hardening through keystroke dynamics. In *Software Reliability Engineering, 2006. ISSRE'06. 17th International Symposium on*, pages 117–126. IEEE, 2006. 5, 19, 44, 58, 64

[7] Luciano Bello, Maximiliano Bertacchini, Carlos Benitez, Juan Carlos Pizzoni, and Marcelo Cipriano. Collection and publication of a fixed text keystroke dynamics dataset. In *XVI Congreso Argentino de Ciencias de la Computación*, 2010. 65

[8] Francesco Bergadano, Daniele Gunetti, and Claudia Picardi. User authentication through keystroke dynamics. *ACM Transactions on Information and System Security (TISSEC)*, 5(4):367–397, 2002. 6, 7

[9] Saleh Bleha, Charles Slivinsky, and Bassam Hussien. Computer-access security systems using keystroke dynamics. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 12(12):1217–1222, 1990. 4

[10] Cheng Bo, Lan Zhang, and Xiang-Yang Li. Silentsense: Silent user identification via dynamics of touch and movement behavioral biometrics. *arXiv preprint arXiv:1309.0073*, 2013. 10, 11, 17, 95

[11] Jürgen Bortz, Gustav Adolf Lienert, and Klaus Boehnke. *Verteilungsfreie methoden in der biostatistik*. Springer DE, 2000. 34, 62

[12] Patrick Bours and Christopher Johnsrud Fullu. A login system using mouse dynamics. In *Intelligent Information Hiding and Multimedia Signal Processing, 2009. IIH-MSP'09. Fifth International Conference on*, pages 1072–1077. IEEE, 2009. 10, 11, 13

[13] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001. 28, 45

[14] P. Campisi, E. Maiorana, M. Lo Bosco, and A Neri. User authentication using keystroke dynamics for cellular phones. *Signal Processing, IET*, 3(4):333–341, July 2009. 95

[15] Wendy Chen and Weide Chang. Applying hidden markov models to keystroke pattern analysis for password verification. In *Information Reuse and Integration, 2004. IRI 2004. Proceedings of the 2004 IEEE International Conference on*, pages 467–474. IEEE, 2004. 5

[16] Tai-Hoon Cho. Pattern classification methods for keystroke analysis. In *SICE-ICASE, 2006. International Joint Conference*, pages 3812–3815. IEEE, 2006. 5

[17] Nathan Luke Clarke and SM Furnell. Advanced user authentication for mobile devices. *computers & security*, 26(2):109–119, 2007. 95

[18] A. Crenshaw. Changing your mac address in window xp/vista, linux and mac os x, 2009. 4

[19] Alexander De Luca, Alina Hang, Frederik Brudy, Christian Lindner, and Heinrich Hussmann. Touch me once and i know it's you!: implicit authentication based on touch screen patterns. In *Proceedings of the 2012 ACM annual conference on Human Factors in Computing Systems*, pages 987–996. ACM, 2012. 15

[20] Paul H Dietz, Benjamin Eidelson, Jonathan Westhues, and Steven Bathiche. A practical pressure sensitive computer keyboard. In *Proceedings of the 22nd annual ACM symposium on User interface software and technology*, pages 55–58. ACM, 2009. 5

[21] Rick Downs. Using resistive touch screens for human/machine interface. *Analog Applications Journal, Texas Instruments*, 2005. 15

[22] Encyclopaedia Britannica Online Academic Edition. Student's t-test, 2014. 81

[23] Tao Feng, Ziyi Liu, Kyeong-An Kwon, Weidong Shi, Bogdan Carbunar, Yifei Jiang, and Nhung Nguyen. Continuous mobile authentication using touchscreen gestures. In *Homeland Security (HST), 2012 IEEE Conference on Technologies for*, pages 451–456. IEEE, 2012. 16, 18, 82, 94

[24] Mario Frank, Ralf Biedert, Eugene Ma, Ivan Martinovic, and Dawn Song. Touchalytics: On the applicability of touchscreen input as a behavioral biometric for continuous authentication. *Information Forensics and Security, IEEE Transactions on*, 8(1):136–148, 2013. 16, 18, 72, 82, 94

[25] Milton Friedman. The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the American Statistical Association*, 32(200):675–701, 1937. 32, 61

[26] Hugo Gamboa and Ana Fred. A behavioral biometric system based on human-computer interaction. In *Defense and Security*, pages 381–392. International Society for Optics and Photonics, 2004. 12

[27] Gartner. Market share analysis: Mobile phones, worldwide, q2 2013, 2013. 69

[28] Romain Giot, Mohamad El-Abed, and Christophe Rosenberger. Greyc keystroke: a benchmark for keystroke dynamics biometric systems. In *Biometrics: Theory, Applications, and Systems, 2009. BTAS'09. IEEE 3rd International Conference on*, pages 1–6. IEEE, 2009. 65

[29] Cristiano Giuffrida, Kamil Majdanik, Mauro Conti, and Herbert Bos. I sensed it was you: Authenticating mobile users with sensor-enhanced keystroke dynamics. In *Detection of Intrusions and Malware, and Vulnerability Assessment*, pages 92–111. Springer, 2014. 95

[30] Joaquin Gonzalez-Rodriguez. Evaluating automatic speaker recognition systems: An overview of the nist speaker recognition evaluations (1996-2014). *Loquens*, 1(1):e007, 2014. ix, 76

[31] Ralph Gross and Alessandro Acquisti. Information revelation and privacy in online social networks. In *Proceedings of the 2005 ACM workshop on Privacy in the electronic society*, pages 71–80. ACM, 2005. 4

[32] Daniele Gunetti and Claudia Picardi. Keystroke analysis of free text. *ACM Transactions on Information and System Security (TISSEC)*, 8(3):312–347, 2005. 6, 7

[33] Shivani Hashia. *Authentication by mouse movements*. PhD thesis, San Jose State University, 2004. 13

[34] Ali Hasimah, Wahyudi Martono, and Salami Momoh Jimoh E. Keystroke pressure based typing biometrics authentication system by combining ann and anfis-based classifiers. 2009. 5

[35] Sylvain Hocquet, J Ramel, and Hubert Cardot. Estimation of user specific parameters in one-class problems. In *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on*, volume 4, pages 449–452. IEEE, 2006. 5

[36] Danoush Hosseinzadeh, Sridhar Krishnan, and April Khademi. Keystroke identification based on gaussian mixture models. In *Acoustics, Speech and Signal Processing, 2006. ICASSP 2006 Proceedings. 2006 IEEE International Conference on*, volume 3, pages III–III. IEEE, 2006. 5

[37] Jiankun Hu, Don Gingrich, and Andy Sentosa. A k-nearest neighbor approach for user authentication through biometric keystroke dynamics. In *Communications, 2008. ICC'08. IEEE International Conference on*, pages 1556–1560. IEEE, 2008. 5

[38] Seong-seob Hwang, Sungzoon Cho, and Sunghoon Park. Keystroke dynamics-based authentication for mobile devices. *Computers & Security*, 28(1):85–93, 2009. 95

[39] ISO/IEC-9995-3:2010. Information technology - keyboard layouts for text and office systems - part 3: Complementary layouts of the alphanumeric zone of the alphanumeric section, 2010. 53, 96

[40] JISX-6002:1980. Keyboard layout for information processing using the jis 7 bit coded character set, 1988. 53, 96

[41] Zach Jorgensen and Ting Yu. On mouse dynamics as a behavioral biometric for authentication. In *Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security*, pages 476–482. ACM, 2011. 12

[42] Rick Joyce and Gopal Gupta. Identity authentication based on keystroke latencies. *Communications of the ACM*, 33(2):168–176, 1990. 5

[43] Pilsung Kang, Seong-seob Hwang, and Sungzoon Cho. Continual retraining of keystroke dynamics based authenticator. In *Advances in Biometrics*, pages 1203–1211. Springer, 2007. 5

[44] Kevin Killourhy and Roy Maxion. Why did my detector do that?! In *Recent Advances in Intrusion Detection*, pages 256–276. Springer, 2010. 5

[45] Kevin S Killourhy. A scientific understanding of keystroke dynamics. Technical report, DTIC Document, 2012. 5

[46] Kevin S Killourhy and Roy A Maxion. Comparing anomaly-detection algorithms for keystroke dynamics. In *Dependable Systems & Networks, 2009. DSN'09. IEEE/IFIP International Conference on*, pages 125–134. IEEE, 2009. x, 5, 6, 22, 42, 43, 64, 65

[47] Lingjun Li, Xinxin Zhao, and Guoliang Xue. Unobservable reauthentication for smart phones. In *Proceedings of the 20th Network and Distributed System Security Symposium, NDSS*, volume 13, 2013. 16, 18

[48] Chien-Cheng Lin, Chin-Chun Chang, and Deron Liang. A new non-intrusive authentication approach for data protection based on mouse dynamics. In *Biometrics and Security Technologies (ISBAST), 2012 International Symposium on*, pages 9–14. IEEE, 2012. 10, 11

[49] Chen Change Loy, Weng Kin Lai, and Chee Peng Lim. Keystroke patterns classification using the artmap-fd neural network. In *Intelligent Information Hiding and Multimedia Signal Processing, 2007. IIHMSP 2007. Third International Conference on*, volume 1, pages 61–64. IEEE, 2007. 5

[50] TouchType Ltd. Swiftkey, 2014. 96

[51] Emanuele Maiorana, Patrizio Campisi, Noelia González-Carballo, and Alessandro Neri. Keystroke dynamics authentication for mobile phones. In *Proceedings of the 2011 ACM Symposium on Applied Computing*, pages 21–26. ACM, 2011. 95

[52] Leenesh Kumar Maisuria, Cheng Soon Ong, and Weng Kin Lai. A comparison of artificial neural networks and cluster analysis for typing biometrics authentication. In *Neural Networks, 1999. IJCNN'99. International Joint Conference on*, volume 5, pages 3295–3299. IEEE, 1999. 5, 11

[53] Microsoft. Windows keyboard layouts, 2014. 54

[54] Vision Mobile. Developer economics q3 2013 state of the developer nation, 2013. 69

[55] Fabian Monrose and Aviel Rubin. Authentication via keystroke dynamics. In *Proceedings of the 4th ACM conference on Computer and communications security*, pages 48–56. ACM, 1997. 6

[56] Jugurta Montalvao, Carlos Augusto S Almeida, and Eduardo O Freire. Equalization of keystroke timing histograms for improved identification performance. In *Telecommunications Symposium, 2006 International*, pages 560–565. IEEE, 2006. 64, 65

[57] Youssef Nakkabi, Issa Traoré, and Ahmed Awad E Ahmed. Improving mouse dynamics biometric performance using variance reduction via extractors with separate features. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, 40(6):1345–1353, 2010. 11, 13, 14

[58] Hidetoshi Nonaka and Masahito Kurihara. Sensing pressure for authentication system using keystroke dynamics. In *International Conference on Computational Intelligence*, pages 19–22. Istanbul, 2004. 5

[59] Nuance. Swype, 2014. 96

[60] Mohammad S Obaidat and David T Macchiarolo. An online neural network system for computer access security. *Industrial Electronics, IEEE Transactions on*, 40(2):235–242, 1993. 5

[61] Mohammad S Obaidat and Balqies Sadoun. Verification of computer users using keystroke dynamics. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 27(2):261–269, 1997. 5, 6

[62] Arun Ross and Anil Jain. Biometric sensor interoperability: A case study in fingerprints. In *Biometric Authentication*, pages 134–145. Springer, 2004. 4

[63] Hataichanok Saevanee, Nathan Clarke, Steven Furnell, and Valerio Biscione. Text-based active authentication for mobile devices. In *ICT Systems Security and Privacy Protection*, pages 99–112. Springer, 2014. 95

[64] Bassam Sayed, Issa Traore, Isaac Woungang, and Mohammad S Obaidat. Biometric authentication using mouse gesture dynamics. 2013. 10, 11, 13

[65] Douglas A Schulz. Mouse curve biometrics. In *Biometric Consortium Conference, 2006 Biometrics Symposium: Special Session on Research at the*, pages 1–6. IEEE, 2006. 11, 13

[66] Android SDK. Get the android sdk, 2013. 69

[67] Abdul Serwadda, Vir V Phoha, and Zibo Wang. Which verifiers work?: A benchmark evaluation of touch-based authentication algorithms. In *Biometrics: Theory, Applications and Systems (BTAS), 2013 IEEE Sixth International Conference on*, pages 1–8. IEEE, 2013. x, 16, 18, 77, 82, 94, 100

[68] Chao Shen, Zhongmin Cai, and Xiaohong Guan. Continuous authentication for mouse dynamics: A pattern-growth approach. In *Dependable Systems and Networks (DSN), 2012 42nd Annual IEEE/IFIP International Conference on*, pages 1–12. IEEE, 2012. 11, 13

[69] Chao Shen, Zhongmin Cai, Roy A Maxion, Guang Xiang, and Xiaohong Guan. Comparing classification algorithm for mouse dynamics based user identification. In *Biometrics: Theory, Applications and Systems (BTAS), 2012 IEEE Fifth International Conference on*, pages 61–66. IEEE, 2012. 12

[70] SJ Shepherd. Continuous authentication by analysis of keyboard typing characteristics. 1995. 5, 7, 53

[71] Terence Sim and Rajkumar Janakiraman. Are digraphs good for free-text keystroke dynamics? In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, pages 1–6. IEEE, 2007. 6, 7

[72] Ki-seok Sung and Sungzoon Cho. Ga svm wrapper ensemble for keystroke dynamics authentication. In *Advances in Biometrics*, pages 654–660. Springer, 2005. 5

[73] Zahid Syed, Sean Banerjee, Qi Cheng, and Bojan Cukic. Effects of user habituation in keystroke dynamics on password security policy. In *High-Assurance Systems Engineering (HASE), 2011 IEEE 13th International Symposium on*, pages 352–359. IEEE, 2011. 5, 12, 61, 62, 64

[74] Issa Traore, Isaac Woungang, Mohammad S Obaidat, Youssef Nakkabi, and Iris Lai. Combining mouse and keystroke dynamics biometrics for risk-based authentication in web environments. In *Digital Home (ICDH), 2012 Fourth International Conference on*, pages 138–145. IEEE, 2012. 14

[75] M. Trojahn and F. Ortmeier. Toward mobile authentication with keystroke dynamics on mobile phones and tablets. In *Advanced Information Networking and Applications Workshops (WAINA), 2013 27th International Conference on*, pages 697–702, March 2013. 96

[76] David Umphress and Glen Williams. Identity verification through keyboard characteristics. *International journal of man-machine studies*, 23(3):263–273, 1985. 5, 7, 11

[77] Mary Villani, Charles Tappert, Giang Ngo, Justin Simone, H St Fort, and Sung-Hyuk Cha. Keystroke biometric recognition studies on long-text input under ideal and application-oriented conditions. In *Computer Vision and Pattern Recognition Workshop, 2006. CVPRW'06. Conference on*, pages 39–39. IEEE, 2006. 7

[78] Kim-Phuong L Vu, Abhilasha Bhargav, and Robert W Proctor. Imposing password restrictions for multiple accounts: Impact on generation and recall of passwords. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, volume 47, pages 1331–1335. SAGE Publications, 2003. 4

[79] Matt Wolff. Behavioral biometric identification on mobile devices. In *Foundations of Augmented Cognition*, pages 783–791. Springer, 2013. 95

[80] J.R. Young and R.W. Hammon. Method and apparatus for verifying an individual's identity, February 14 1989. US Patent 4,805,222. 4

[81] Enzhe Yu and Sungzoon Cho. Keystroke dynamics identity verification-its problems and practical solutions. *Computers & Security*, 23(5):428–440, 2004. 5, 11